Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL: **Control mediante tecnología IOT "low power" de variables relacionadas con un proceso industrial.**

AUTOR: DEU ALMOR, MARC

DATA DE PRESENTACIÓ: 30 Octubre, 2019

COGNOMS: Deu Almor **NOM: Marc**

TITULACIÓ: Grau en Enginyeria Electrónica Industrial i Automàtica

PLA:

DIRECTOR: Antonio Miguel López Martínez

DEPARTAMENT:

QUALIFICACIÓ DEL TFG

TRIBUNAL

PRESIDENT **SECRETARI** **VOCAL**

DATA DE LECTURA:

Aquest Projecte té en compte aspectes mediambientals: □ Sí □ No

RESUM

La revolución 4.0 ha llegado y con ella están apareciendo gran cantidad de posibles aplicaciones de proyectos de IOT. La necesidad de controlar mediante sensores los procesos productivos conlleva la aparición de un gran mercado de posibilidades. Además, si estos módulos son de fácil instalación y autónomos todavía incrementa más sus posibilidades.

Este proyecto pretende mostrar los conocimientos básicos para poder llevar a cabo un proyecto de monitorización y control de un proceso industrial, empezando por el marco teórico en el que se incluyen las diferentes tecnologías de comunicación existentes, los diversos microprocesadores capaces de procesar los datos y una pequeña muestra de sensores para la toma de estos.

Todo ello implementado en un ejemplo en el que se intentan solventar las principales casuísticas de comunicación y procesado de datos, además de la realización de una pequeña interfaz para el control de estos datos.

Como resultado se obtiene un conjunto de datos que puede utilizarse como la base de proyectos de IOT de bajo consumo más complejos, de implementación real en la industria.

Paraules clau (màxim 10):

Industria 4.0	Low Power	ZigBee	Xbee S1 PRO
Sensores	Datos	Monitorización	Control
Inyección plásticos	Red		

ABSTRACT

4.0 industrial revolution have arrived, and with it the number of IOT applications have increased a lot. The grown necessity of controlling, with the use of sensors, industrial process produces the appearance of a new market of possibilities. In addition, if these nodes have a simple mode of installation, their possibility’s increase even more.

This project tries to show the basic concepts for the implementation of a control and monitoring project in an industrial environment. Staring with the theorical base in which there are included the principal wireless low power technologies, the different type of data processors and a little sample of different data sensors.

All this base implemented in an example, in which I tried to implement the principal communication and data processing casuistics. Also, the realization of a user interface to control this process.

The result is the obtention of a data packet, that could be used as a base mark for complex industrial IOT low power projects.

Keywords (10 maximum):

4.0 Industry	Low Power	Zigbee	Xbee
Sensing	Data analysis	Monitorization	Control
Injection machines	Network		

Sumario

SUMARIO DE FIGURAS

GLOSSARI DE SIGNES, SÍMBOLS, ABREVIATURES, ACRÒNIMS I TERMES

ZC Zigbee Coordinator
ZR Zigbee Router
ZED Zigbee end device
PAN Personal Area Network
DSSS Direct sequence spread spectrum
FHSS Frequency hopping spread spectrum
IOT “internet off things” Internet de las cosas
LTE-M Long Term Evolution for Machines
NB-IoT NarrowBand IoT
LPWA Low Power Wide Area
CSS Chirp Spread Spectrum
OSCI Open Systems Interconection
CSMA-CA Collision avoidance techniques

INTRODUCCION

La aparición del nuevo concepto de industria 4.0 supone una nueva manera de organizar los medios de producción y mejorar la adaptabilidad de las plantas industriales hacia una nueva revolución industrial, la cuarta revolución industrial.

Dentro de las bases tecnológicas de la industria 4.0 aparece el concepto de IoT, internet de las cosas, bajo este concepto se pretende la interconexión de todos los dispositivos de un proceso mediante una red donde todos ellos podrían ser visibles e interactuar.

En este trabajo se propone la realización de un estudio de las redes IOT y las tecnologías más usuales, así como los módulos que existen en el mercado y sus posibilidades. Para a continuación implementar estos conocimientos en un caso de ámbito industrial en la monitorización y control de los parámetros de una planta de inyección de plásticos.

Dentro de los módulos IoT, se priorizará crear una configuración de bajo consumo, además de estudiar métodos de optimización de la vida útil de los dispositivos mediante: modos de sueño, modos de bajo consumo y técnicas de recolección de la energía, también conocidas como “Power harvesting techniques”.

Además, se analizará la posibilidad de que este dispositivo sea de tipo “plug and play”, facilitando así su aplicación en el entorno industrial.

Por último, se aplicará un modelo prototipo en una pequeña muestra de una planta de inyección de plásticos, como primera fase de proyecto para obtener resultados y poder mejorar el producto de cara a una implementación y comercialización de este producto.

1.1 OBJETIVOS

Los objetivos principales de este proyecto son: el estudio de las tecnologías de comunicación inalámbrica de bajo consumo y su implementación en el control de un proceso industrial.
Este objetivo se aplicará en el control y monitorización de los principales parámetros de una planta de inyección de plásticos.

Además de este objetivo principal, algunos objetivos secundarios que derivan de este son el estudio de las diferentes topologías de red, la comparativa entre los diferentes protocolos de comunicación inalámbrica radio en bandas no licenciadas, el estudio de las baterías, como reducir el consumo de mi aplicación para mejorar su periodo de vida y por último, las técnicas de recolección de energía “power harvesting techniques”.

Todos estos objetivos concluirán en la creación de unos dispositivos electrónicos de monitorización y control para una planta de inyección de plásticos. Estos dispositivos podrán ser posicionados en cualquier parte de la máquina para, con la ayuda de sensores, enviar información de estos a un coordinador. El coordinador será el encargado de controlar, con la información que reciba, el correcto funcionamiento de las máquinas y sus periféricos de una manera totalmente autónoma, además de tener un periodo de trabajo útil superior al año de manera ininterrumpida.

Este dispositivo electrónico será de fácil instalación i seguirá el principio “plug and play”.

1.2 MOTIVACION

Actualmente estamos inmersos en un boom tecnológico en aquello que conocemos por el término de industria 4.0. Una de las áreas de estudio dentro de este concepto es el IOT, que entre sus aplicaciones engloba la monitorización de plantas de producción para su optimización y control, implementando, a ser posible, protocolos de comunicación sin cables, de bajo consumo.

Mi motivación para llevar a cabo este proyecto está basada en tres hechos principales. En primer lugar, mi curiosidad por este tipo de dispositivos de bajo consumo y la gran cantidad de aplicaciones y soluciones que pueden aportar a la industria, en segundo lugar, una necesidad real en mi puesto de trabajo de controlar la producción de la sala de inyección de plásticos y el óptimo funcionamiento de las máquinas y por último, pero no menos importante, el hecho de que dispongo de un profesor en la universidad especializado en este área.

2. ESTADO DEL ARTE

En este apartado se estudia la situación de las tecnologías de red IoT que existen en el mercado.

2.1 COMUNICACIONES SIN CABLES

Dentro de los protocolos de comunicación sin cables existen dos grandes grupos: los protocolos de monitorización continua y los de monitorización en intervalos. Estos intervalos pueden ser o no regulares.
Podemos hacer otra diferenciación dentro de aquellos protocolos de monitorización en intervalos entre los protocolos pensados para IOT i los que no.

Estos protocolos pensados para IOT se caracterizan por largo alcance de comunicación, velocidades de trasmisión de datos bajas, así como baja frecuencia de transmisión. Dentro de este grupo se puede diferenciar entre aquellos protocolos que trabajan en bandas no licenciadas como son ZigBee, Lora, SigFox. Y aquellos que lo hacen en bandas licenciadas como LTE-M o NB-IoT.

2.1.1 LTE-M

LTE-M es un término industrial simplificado de LTE-MTC LPWA. Este protocolo de comunicación móvil conecta a internet dispositivos reutilizando la red existente LTE, es decir, 4G y hereda ventajas de esta red como es la seguridad o la privacidad de la información.

Figure 1. Logo LTE-M

Las principales ventajas de este protocolo es que soporta dispositivos en movilidad, permite cobertura en interiores, su ancho de banda facilita el envio de imágenes o incluso voz y dispone de una baja latencia.

2.1.2 NB-IOT

Narrowband-IOT es un protocolo desarrollado para permitir una comunicación eficiente y una larga duración de la batería para dispositivos distribuidos de manera masiva, mediante la utilización de la red de telefonía móvil.

Las tres características que diferencian este protocolo de otros en banda licenciada es su bajo coste, de alrededor de unos 8 euros por dispositivo; la capacidad de conectar un número mayor de dispositivos por antena, ya que al utilizar un ancho de banda de 180KHz aproximadamente cada red puede albergar unas 100.000 conexiones; y por último la gran capacidad de penetración en interiores y bajo tierra.

Además de un consumo sumamente bajo que permite periodos de vida de las baterías superiores a los 10 años.

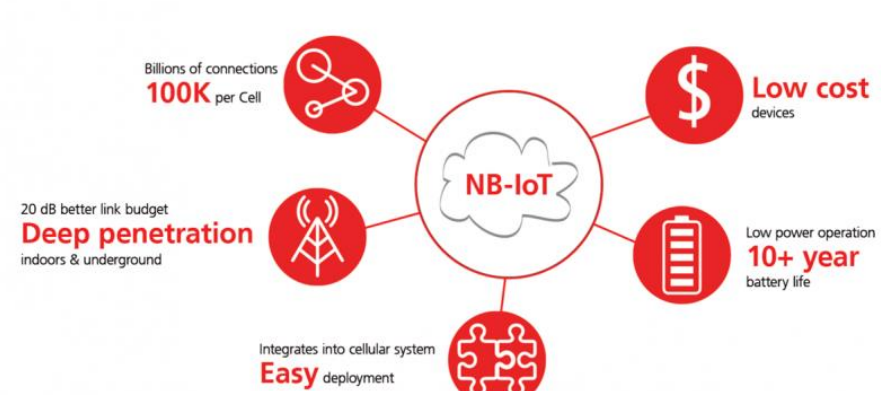


Figure 2. NB-IOT características principales

2.1.2.1 COMPARATIVA ENTRE NB-IOT Y LTE-M

Protocolo	*NB-IOT*	*LTE-M*
Ancho de banda	180KHz 3GPP Licensed	1.4MHz 3GPP Licensed
Velocidad máxima	<100	384Kbps
Velocidad de bajada/subida	27.2 / 62.5 Kbps	1 Mbps
Latencia	1.5 – 10 seg.	50-100 ms
Duración de las baterías	+10años	10 años
Consumo de energía	Inferior a velocidad de datos baja	Inferior a velocidad de datos media
Coste por módulo	4-9 euros	8-14 euros
Despliegue de frecuencia	Flexible	Banda LTE
Penetración en interiores	Excelente	Buena
Transmisión de voz	No	Si

2.1.3 LoRa

LoRa es otro protocolo de red LPWAN que a diferencia de otros protocolos utiliza tipo de modulación en radiofrecuencia propio llamado Chirp Spread Spectrum (CSS). Este tipo de modulación está pensado para comunicaciones a largas distancia con una baja influencia de interferencias en los paquetes de información de hasta 242 bytes.

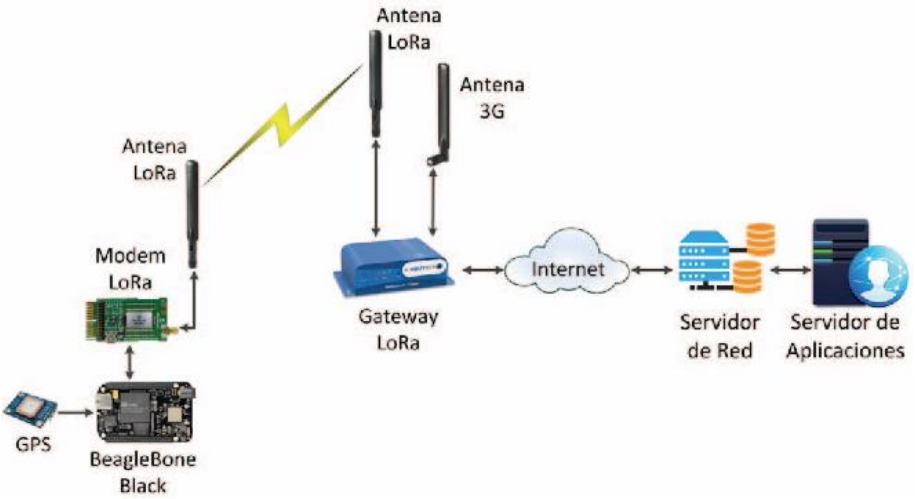


Figure 3. Principal configuración de red LoRa

A diferencia de otros protocolos LoRa funciona muy bien en comunicaciones bidireccionales, aunque la arquitectura entre nodos siempre será en modo estrella. Esto puede ser un problema en según qué aplicaciones de IOT.

LoRa es un protocolo de banda libre i utiliza la banda ISM de frecuencias para enviar datos, en el caso de Europa la banda de 868MHz.

Como podemos observar en la imagen la arquitectura de funcionamiento de LoRa son un en estrella envían datos a un “Gateway” i este ya sea via Ethernet, Wi-fi o conexión móvil lo envía a los servidores de datos, de donde toman los datos las aplicaciones de IOT de la red.

2.1.4 SIGFOX

Sigfox como los demás es un protocolo LPWAN creado por una empresa con este mismo nombre en 2009. Está pensado para el funcionamiento en módulos de muy bajo consumo con pequeños sensores alimentados normalmente por pilas. Los paquetes de información que es capaz de enviar son de hasta 12 Bytes.
La arquitectura de nodos de SigFox permite que todos los módulos son routers y por tanto la información puede ser enviada o recibida desde la nube a cualquier nodo.

Igual que LoRa también utiliza bandas no licenciadas ISM.

En lo que a seguridad se refiere, LoRa asigna un código de identificación a cada nodo y además cuenta con protocolos de encriptamiento VPN, implementando al final el protocolo https.

Figure 4. Configuración de red SigFox

2.1.5 ZIGBEE

Zigbee es un protocolo de comunicación inalámbrico basado en el estándar de comunicaciones, IEEE_802.15.4, para redes inalámbricas. Este protocolo de comunicación está desarrollado por Zigbee Alliance, que a su vez está formado por algunas de las empresas más innovadoras y tecnológicas del mundo. Algunos de sus integrantes son: Shneider Electric, Huawei, Amazon, Comcast entre otros.

Figure 5. Bases de Zigbee

La tecnología zigbee está enfocada a aplicaciones de Iot “Internet of things”, y una de sus principales diferencias respecto a otros sistemas de comunicación inalámbrica es la posibilidad de generar redes de miles de dispositivos, hasta un límite de 65535 equipos, que interconectados en diferentes niveles transmitan datos entre ellos de los diferentes nodos de la red.

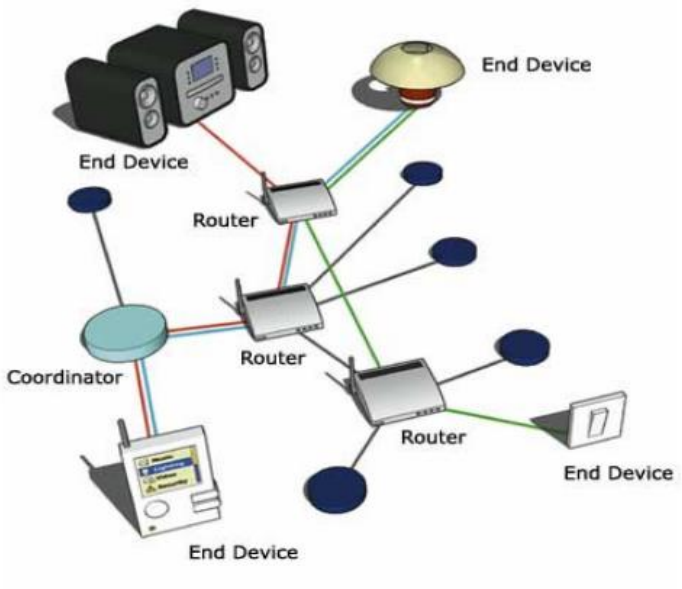


Figure 6. Red de comunicaciones Zigbee

Este protocolo de comunicación está pensado para que dispositivos de bajo consumo puedan intercomunicarse inalámbricamente. Por ejemplo, para usos domóticos o de sensórica en entornos industriales.

Las especificaciones básicas de ZigBee son una velocidad desde 250Kbps a 2.4Ghz. Esta velocidad puede parecer lenta a primera vista comparándola con la de otros protocolos similares de conexión sin cables como bluetooth o wi-fi. Pero la velocidad no es lo más importante para ZigBee ya que este a diferencia de los demás protocolos mencionados está pensado para monitorizar paquetes de información en intervalos no regulares manteniendo un consumo bajo.

Algunas de las características principales de este protocolo de comunicación de bajo consumo son las siguientes:

- Muy bajo consumo eléctrico
- Bajo coste
- Muy flexible y apto para grandes redes de comunicación
- Uso de una banda de radio no licenciada

El protocolo ZigBee puede trabajar en diferentes configuraciones como: la sustitución de un cable de puerto serie sin cables, una comunicación punto a punto, una comunicación multipunto o desde comunicaciones “peer to peer” a configuraciones de red más complejas.

2.1.6 ZIGBEE STACK LAYERS

El protocolo Zigbee está compuesto por 4 capas. Está basado en el modelo “Open Systems Interconection” OSCI, y aunque este modelo se compone de 7 capas Zigbee simplifica el suyo a solo 4.

A continuación, podemos observar las diferentes capas del protocolo Zigbee y como se comunican entre ellas y podemos apreciar que Zigbee define únicamente las dos capas de red (APS y NWK) mientras que toma del estándar IEEE 802.15.4 la capa MAC y la PHY. [1]

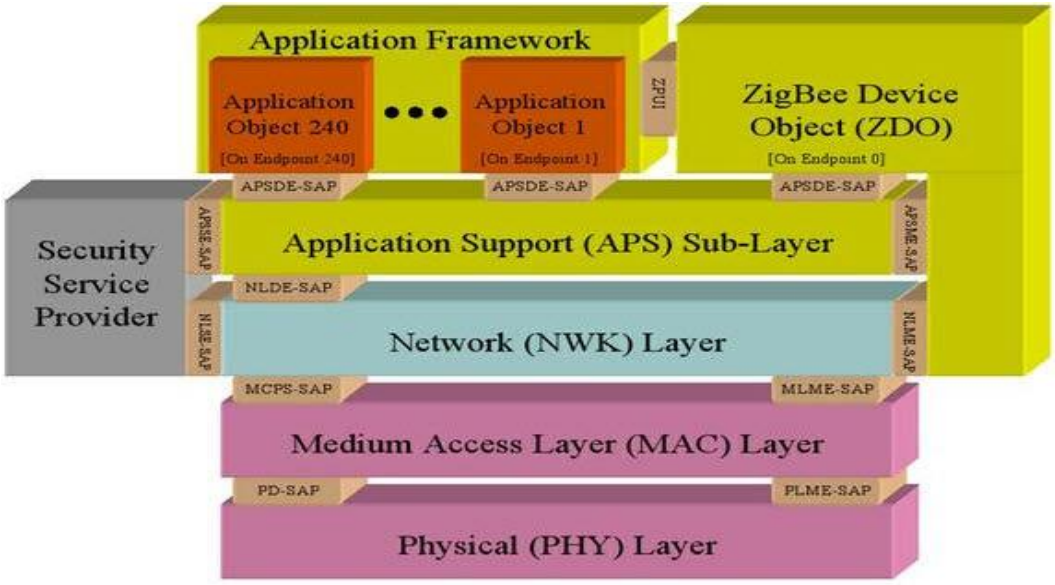


Figure 7.-Capas de la tecnología Zigbee

2.1.6.1 PHYSICAL LAYER

Esta capa de red es la encargada de definir las operaciones físicas del dispositivo zigbee. Incluye la sensibilidad de recepción, el voltaje de salida, el número de canales, la modulación del chip y las especificaciones de transmisión.

2.1.6.2 MEDIUM ACCESS LAYER

Esta capa de la red es la encargada de controlar las transacciones de datos vía radio frecuencia entre nodos cercanos, en comunicaciones punto a punto. Además incluye servicios de verificación de la transmisión y de reintento de envío. También incluye las técnicas encargadas de evitar colisiones (CSMA-CA).

2.1.6.3 NETWORK LAYER

Esta capa de red es la encargada de añadir capacidad de enrutamiento, que permiten a los paquetes de radio frecuencia moverse entre diversos nodos para llegar de un punto inicial a su destino.

2.1.6.4 APPLICATION SUPPORT SUB-LAYER

Esta capa de red es la encargada de definir las direcciones de los diversos objetos nodos del sistema red.

2.1.6.5 APLICATION FRAMEWORK

Esta capa de red provee las capacidades de manejo de red avanzadas, así como la capacidad de encontrar nodos entre ellos.

2.1.7 LOS ELEMENTOS PRINCIPALES DE UNA RED ZIGBEE

Existen tres configuraciones básicas de los nodos en las redes de dispositivos Zigbee. Un coordinador, algunos routers y nodos finales.

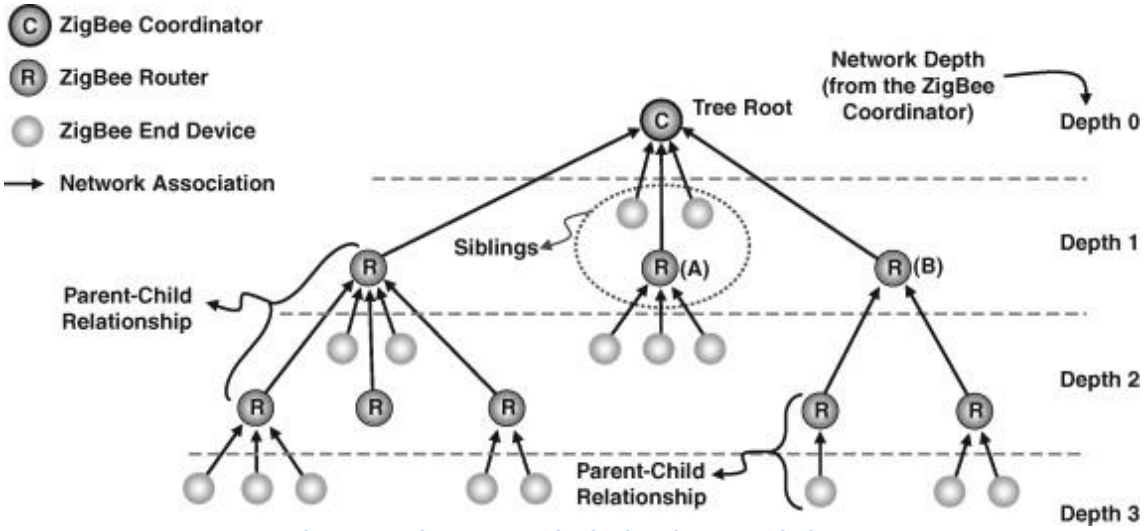


Figure 8.- Elementos principales de una red ZigBee

- Coordinador (ZC)

El coordinador Zigbee coordina la red como un conjunto. Su función principal es la creación i parametrización de la red. Es el encargado de elegir el identificador PAN y el canal de radio en el que operará la red. Una vez creada y configurada la red el nodo coordinador pasa a desarrollar un trabajo de router.
Es importante mencionar que en toda red solo puede haber un coordinador, y que este puede enviar y recibir ya sea a otros routers o a otros nodos finales.

- Router (ZR)

El router Zigbee controla la distribución de la red. Su función principal es encontrar el mejor camino para enviar paquetes de información desde un nodo a otro.
El router actúa como el punto de comunicación entre otros nodos ya sean otros routers o nodos finales.

- Nodo Final (ZED)

Los nodos finales de Zigbee pueden establecer comunicación con routers o el coordinador, pero no pueden comunicarse con otros nodos finales.
Una importante característica de estos nodos, es que su consumo eléctrico es muy bajo y por tanto son apropiados para aplicaciones de bajo consumo en las que el nodo final solo consume cuando ocurre un evento que lo despierta.

2.1.8 TOPOLOGÍAS PRINCIPALES DE UNA RED ZIGBEE

De acuerdo con el protocolo Zigbee estándar existen tres tipos principales de topologías de red: árbol, estrella y red de malla o mesh.

Es importante mencionar que en cualquier topología de red zigbee existe el nodo coordinador encargado de crear la red y centralizar la adquisición de datos.

La primera topología de red es la configuración en estrella, en la que un coordinador zigbee se conecta con diferentes nodos finales.

La segunda topología de red incluye dos topologías de conexión punto a punto. La topología tipo malla, en la que un coordinador está conectado con varios routers y estos están conectados entre ellos y a algunos nodos finales.

Por último, la segunda topología de este segundo grupo es la arquitectura en árbol, en la que un coordinador conecta con varios routers, que no están conectados entre ellos, y estos routers se conectan cada uno con algún nodo final.

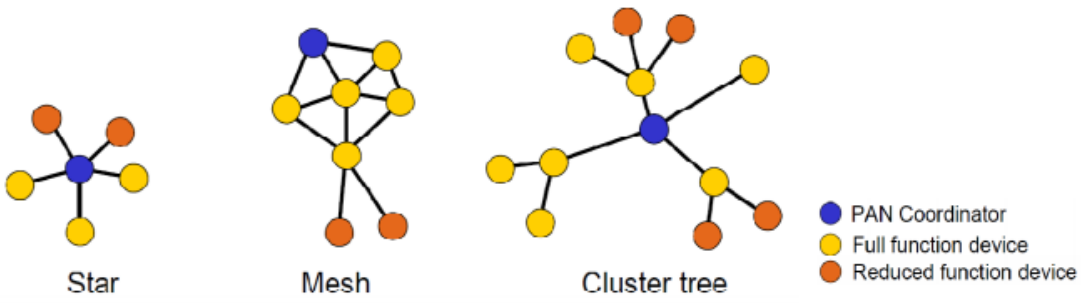


Figure 9.- Topologías de una red Zigbee

2.1.9 COMPARACIÓN ENTRE TRES ESTÁNDARES DE CONEXIÓN SIN CABLES

Este apartado compara dos de los estándares principales de conexiones sin cables con el protocolo Zigbee.

Estos tres estándares trabajan en la misma banda de frecuencias de 2.4GHz, y se utilizan para aplicaciones muy diversas, normalmente uno de los factores que diferencian la utilización de uno de estos protocolos o de otro es la diferencia de requerimientos de consumo eléctrico y el rango de conexión.

Esta comparación se establece entre los protocolos Bluetooth, Wi-fi, ZigBee.

Comparación de protocolos de conexión sin cables

Standard	**Zigbee**	**Bluetooth**	**Wi-Fi**
MAC	IEEE 802.15.4	IEEE 802.15.1	IEEE 802.11b
Range	Indoor: 30m Autdoor: 100m	9m	75m to 90m
Current consumption	25 to 35 mA (Tx/Rx Mode) 3uA (standby mode)	60mA (Tx/Rx mode)	400mA (Tx/Rx mode) 20mA (standby mode)
Data Rate	250Kbps	1Mbps	11Mbps
Stack size	32KB or 4KB possible in ZED	250KB	1MB
Network join time	30ms	More than 3s	1s
Interference method	DSSS	FHSS	DSSS
Minimum Bandwidth	3 MHz (Static)	15 MHz (Dynamic)	22 MHz (Static)
Maximum number of nodes per network	64K	7	32 per access point
Number of channels	16	19	13

2.1.10 SEGURIDAD EN UNA RED INALAMBRICA

La Seguridad es uno de los puntos fuertes del protocolo ZigBee, que implementa un modelo de seguridad basado en el definido por el protocolo IEE 802.15.4.

Este protocolo permite desde comprobación de acceso de los dispositivos a la red, como cifrado utilizando criptografía de clave simétrica. También utiliza comprobación de integridad mediante protocolos MIC.

Zigbee utiliza 3 tipos distintos de claves según el tipo de conexión que utilizan los dispositivos, según se asocien a una red, a un grupo de dispositivos o el enlace entre dos nodos.

- Clave maestra: Clave a partir de la que se generan las diferentes claves de enlace
- Clave de enlace: cifra las comunicaciones punto a punto entre nodos y varia entre pares de nodos.
- Clave de red: clave de la red conocida y utilizada por todos los elementos de una red.

2.1.10.1 DEBILIDADES DE UNA RED ZIGBEE

La principal debilidad de las redes Zigbee recae en los propios nodos de la red, estos almacenan las claves de red en su propia memoria y esto los hace vulnerables. Ya que se puede acceder a la clave si se tiene acceso físico a alguno de los nodos vía el propio software del nodo. Estos softwares no tienen seguridad en el acceso físico.

Una vez se ha accedido a uno de los nodos ya se tiene acceso a toda la red mediante la propia red.

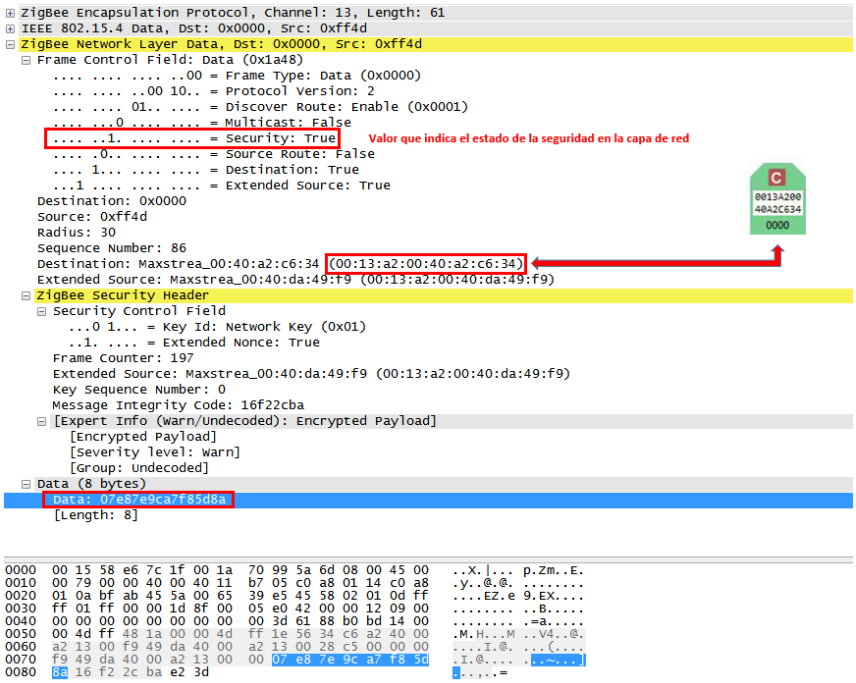


Figure 10.- Respuesta de un nodo final con cifrado

Como conclusión podemos determinar que, aunque el protocolo zigbee dispone de una seguridad de conexiones elevada, las aplicaciones de ésta se ven perjudicadas por la facilidad con la que se puede tener acceso a las claves de red vía acceso físico a uno de los nodos de la red.

Zigbee dispone de características de eficiencia energética, de red y protocolos de seguridad que permiten su aplicación en entornos con altos requerimientos, no obstante, la correcta configuración de los dispositivos es un factor imprescindible para conseguir redes y conexiones seguras.

2.1.11 COEXISTENCIA DE REDES INALÁMBRICAS

En la actualidad disponemos de una gran cantidad de dispositivos interconectados de manera inalámbrica.
Entre ellos podemos destacar la tecnología Wi-fi, bluetooth, 3G y 4G, infrarrojos, además de todo el espectro de radio.

Es por ello que es importante analizar la posibilidad de interferencias en el uso de algunas de estas tecnologías que comparten banda de frecuencias.

En nuestro caso Xbee basado en el protocolo Zigbee, trabaja en una banda de frecuencia de 2.4Ghz que a su vez es utilizada por el protocolo Wi-fi y Bluetooth.

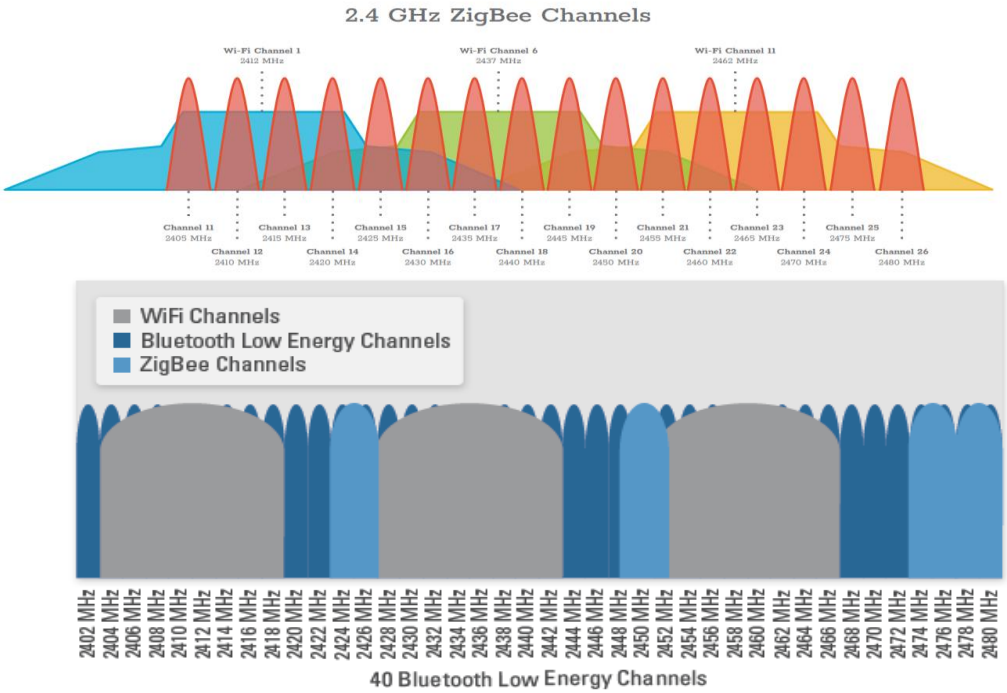


Figure 11.- Coexistencia redes 2.4GHz

Zigbee divide su banda de frecuencias en 16 canales de 2MHz con 5MHz de separación, Wi-fi divide su banda en 14 canales de 22MHz mientras que bluetooth utiliza 79 canales de 1MHZ.
Por tanto, Bluetooth no tiene solapamiento, pero entre wifi y Zigbee si lo hay.
La solución podría ser la selección de canales que no coincidan entre ellos.

2.2 HARDWARE

En esta parte del trabajo se especifica el hardware elegido para la aplicación y su estado del arte

2.3.1 MODULOS XBEE

Los módulos xbee son módulos de comunicación por radio frecuencia, comercializados por la empresa Diji basados en el estándar zigbee, IEE 802.15.4.

Los módulos xbee están diseñados para aplicaciones de comunicación inalámbrica desde la sustitución de un cable de comunicación serie, hasta la creación de redes más complejas tipo “Fast point to multipoint”, es decir redes punto a multipunto, o redes de tipo “peer to peer”, es decir, punto a punto.

Existen una gran variedad de módulos xbee algunos son Zigbee estándar mientras que otros son modificaciones o adaptaciones de este. Además, también existen varias series de producto desde que apareció en el mercado hasta la actualidad, estas series incorporan pequeños cambios y adaptaciones para aplicaciones de cierta complejidad, no obstante, mantienen un modelo con la geometría y la distribución de las pines igual a las series anteriores de tal forma que para aplicaciones sencillas son intercambiables y deberían funcionar correctamente.

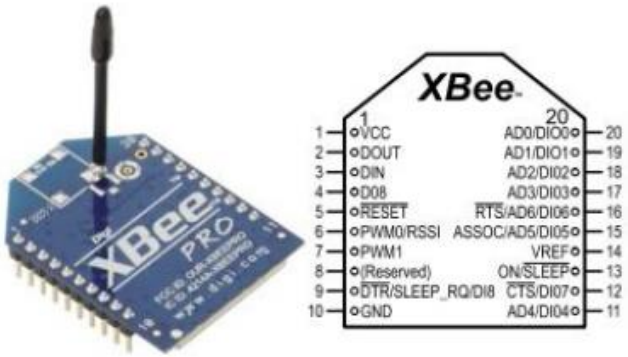


Figure 12.- Modulo Xbee pro serie 1

Entre los módulos xbee encontramos para cada serie de productos la versión estándar y la versión “pro”, cuya diferencia recae básicamente en un augmento del alcance de transmisión de datos.

Entre las series de Xbee podemos encontrar:

2.3.1.1 XBEE SERIE 1

Es la primera serie de productos que sacó al mercado Diji, son la serie más fácil de trabajar, en el caso de comunicaciones punto a punto son mucho más sencillos que las series precedentes ya que carecen del trabajo de preconfiguración.

No obstante, esta ventaja se vuelve un inconveniente de compatibilidad con la serie 2

de productos. Ya que estos si requieren de una preconfiguración y por tanto no hay compatibilidad directa entre ellos.

2.3.1.2 XBEE SERIE 2

La serie dos de diji fue un intento de mejora de la serie 1 que no acabo de salir del todo bien, esta serie se compone de 3 modelos: el xbee Znet 2.5, el ZB y el 2B.

Estos tres modelos son mejoras sucesivas del modelo anterior para lograr una mejora respecto a la serie 1. Y aunque comparten finalidad con la serie 1, mejorando algunos aspectos, la necesidad de preconfigurar los módulos y la mayor complejidad de funcionamiento provocaron una baja aceptación del producto por parte de los usuarios, que seguían apostando por la serie 1 de productos.

Figure 13.- Xbee serie 2

2.3.1.3 XBEE SERIE 3

Esta serie de productos acaba de salir a la luz, y mediante unos costes muy bajos permite realizar conexiones inalámbricas entre dispositivos electrónicos.

La principal mejora respecto a las anteriores series es que permite con un solo módulo conectar con variedad de comunicaciones: BLE, Mesh, Zigbee, 802.15.4.

Además, no requieren de un microcontrolador asociado al módulo ya que se permite la programación del propio microprocesador de xbee para realizar aplicaciones mediante el soporte de microphyton.

2.3.1.4 900 MHz

Figure 14.- Xbee serie 3

Estos módulos no configuran en si una serie, pero si son una familia, es decir, en todas

las series encontramos una versión de los módulos con esta frecuencia tienen una mayor penetración y pueden llegar muy lejos con una antena de alta ganancia.
No obstante, la utilización de esta frecuencia está prohibida en bastantes países y existe una versión de baja frecuencia de 868MHz para solventar este tipo de problemas.

2.3.1.5 XSC

Esta serie de módulos de encuentran dentro de los de 900MHz, son módulos de baja frecuencia que partiendo de los de 900MHz sacrifican parte de la velocidad de datos para aumentar todavía más el alcance.

Para hacernos una idea, la velocidad normal de un módulo de 2.4GHz es de 250Kbps, mientras que las de un módulo de 900MHz es de 156Kbps y la de un módulo XSC es de alrededor de los 10Kbps.

Aunque tengan muy poca velocidad estos módulos disponen de un alcance de hasta 24 Km. y tienen una penetrabilidad muy buena.

2.3.1.6 ANTENAS XBEE

Aunque podemos mejorar el alcance de nuestros dispositivos variando el modelo que escogemos y eligiendo si queremos la versión normal o pro. Un factor muy importante para mejorar el alcance de nuestros nodos es la antena.

Todos los modelos xbee se comercializan con 4 tipos de antenas, antena de chip, whip antena, conector para antenas de tipo U.FL o conector para antenas RSMA.

Las antenas de chip y de tipo whip están más pensadas para aplicaciones electrónicas abiertas mientras que las de conector se adecuan más a la utilización de un dispositivo encapsulado con la antena en el exterior de la caja. Este último tipo de antenas otorga un mejor alcance a nuestros dispositivos que en el caso de ser antenas de alta ganancia puede suponer un alcance de hasta 24 km, suponiendo que el alcance básico del modelo con antena tipo chip regular es de unos 1.5 km en un área abierta sin obstáculos.

Figure 15.- Tipos de antenas Xbee

2.3.2 MODOS DE OPERACIÓN XBEE

Los módulos Xbee pueden operar en 5 modos, para ello es necesario variar el firmware de nuestro dispositivo o activar las opciones que queramos utilizar previamente en la configuración del dispositivo.

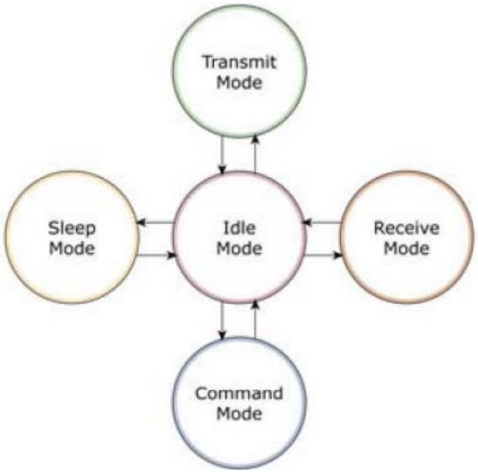


Figure 16.- Modos de operación de Xbee

2.3.2.1 MODO IDLE

Este modo está activo cuando el nodo no envía ni recibe paquetes de datos, es un estado de trámite entre los otros modos de operación. El dispositivo cambiara de modo de operación en el momento que ocurra algún evento.

En el caso de recibir un paquete de datos cambiará a “Recieve mode”, en el caso de tener datos en el buffer cambiará a “transmit mode”, en el caso de cumplir las condiciones de bajo consumo cambiará a “sleep mode” y por último en el caso de entrar un inicio de secuencia de comandos pasará a “comand mode”.

2.3.2.2 MODO TRANSMITIR Y RECIBIR

Se encuentra en este modo cuando le llega algún paquete de datos a través de la antena, o bien envía datos modo serial a través del buffer del pin 3.

La información transmitida puede ser directa o indirecta. La diferencia es ,que en modo directo se envía la información directamente, mientras que en el modo indirecto la información se envía solo cuando el destino la solicita.

La información también se puede enviar de manera “unicast” o “broadcast”. El modo broadcast comunica a todos los nodos de la red para llegar al destinatario y no tiene confirmación de recepción, mientras que el modo unicast envía directamente de un nodo a otro y espera confirmación vía la recepción de un paquete de datos ACK.

2.3.2.3 MODO DE BAJO CONSUMO

Este modo de operación también conocido como “Sleep mode”[3], se activa en el momento en que se cumple alguna de las características siguientes:

Sleep mode setting	Transition into sleep mode	Transition out of sleep mode (wake)	Characteristics	Related commands	Power consumption
Pin hibernate **SM** 1	Assert (high) Sleep_RQ (pin 9)	De-assert (low) Sleep_ RQ	Pin/Host-controlled/NonBeacon systems only/Lowest Power	(**SM**)	< 10 µA (@3.0 VCC)
Pin doze **SM** 2	Assert (high) Sleep_RQ (pin 9)	De-assert (low) Sleep_ RQ	Pin/Host-controlled/NonBeacon systems only/Fastest wake-up	(**SM**)	< 50 µA
Cyclic Sleep **SM** 4	Automatic transition to Sleep Mode as defined by the **SM** (Sleep Mode) and **ST** (Time before Sleep) parameters	Transition occurs after the cyclic sleep time interval elapses. The time interval is defined by the **SP** (Cyclic Sleep Period) parameter.	RF module wakes in pre-determined time intervals to detect if RF data is present/When **SM** = 5	(**SM**), **SP**, **ST**	< 50 µA when sleeping
Cyclic Sleep **SM** 5	Automatic transition to Sleep Mode as defined by the **SM** (Sleep Mode) and **ST** (Time before Sleep) parameters or on a falling edge transition of the SLEEP_RQ pin	Transition occurs after the cyclic sleep time interval elapses. The time interval is defined by the **SP** (Cyclic Sleep Period) parameter.	RF module wakes in pre-determined time intervals to detect if RF data is present. Module also wakes on a falling edge of SLEEP_RQ.	(**SM**), **SP**, **ST**	< 50 µA when sleeping

Figure 17.- Opciones Sleep mode

Tal y como podemos observar en la tabla anterior existen varias opciones de configuración del modo de bajo consumo.

2.3.2.3.1 PIN DE HIBERNACIÓN (SM1)

Se trata del primer modo de sueño de los Xbee, SM1, se activa al poner en nivel alto el pin 9.

En este momento el módulo Xbee finalizará la acción que tenga en curso y pasará a modo reposo y posteriormente a hibernación. Para deshabilitar este modo bastará con deshabilitar el pin 9 y el Xbee volverá a su modo de operación normal.

2.3.2.3.2 PIN 12 (SM2)

El modo de hibernación pin doze, SM2, funciona de manera parecida al modo SM1. Este modo difiere del anterior por tener un tiempo de “despertar” más rápido y en consecuencia un consumo energético mayor.

Pero el funcionamiento de hibernación y despertar es el mismo que en el caso anterior.

2.3.2.3.3 MODO DE SUEÑO CÍCLICO REMOTO (SM4)

Este modo de hibernación funciona cíclicamente, es decir, el nodo revisa su buffer de entrada y salida periódicamente y el resto del tiempo se mantiene dormido. Los ciclos de sueño son configurables.

2.3.2.3.4 MODO DE SUEÑO CÍCLICO CON DESPERTAR POR PIN (SM5)

Este modo de sueño es muy parecido al modo de sueño cíclico SM4, además de tener las funcionalidades del modo anterior añade la posibilidad de despertar el nodo vía el pin 9 o radio frecuencia.

No obstante, mientras el nodo tenga alguna actividad no retornará al modo cíclico de hibernación.

2.3.2.3.5 MODO DE SUEÑO CÍCLICO COORDINADO POR UN NODO (SM6, SM7, SM8)

Este modo de hibernación permite controlar el estado de los nodos por radio frecuencia.

En el caso de SM7 y SM8 son referidos al driver digimesh en el que nos permiten dormir todo un conjunto de nodos de una red vía RF desde un dispositivo coordinador de la hibernación.

En este caso los nodos sincronizan su hibernación al recibir un paquete de datos enviado por el coordinador de la hibernación.

Este modo de sueño es muy útil en sistemas de monitorización de bajo consumo ya que es posible encender la red únicamente para tomar las lecturas de los sensores y acto seguido dormir la red para alargar la vida útil de las baterías de nuestro sistema de bajo consumo.

2.3.2.4 MODO DE COMANDO

Este modo de funcionamiento de Xbee, es un estado en que el firmware interpreta los caracteres entrantes como comandos. Existen dos tipos de comandos en la serie 1 de Xbee. Los comandos AT y los comandos API

2.3.2.4.1 MODO AT

Este modo permite leer y variar la configuración de los nodos via RF. Cada xbee tiene su correspondiente identificador para poder acceder a el vía RF.

La manera de variar un parámetro es enviar una trama compuesta de la siguiente manera:

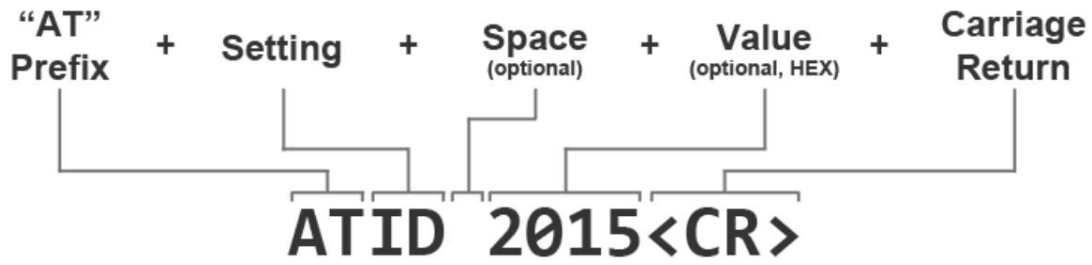


Figure 18.- Trama AT mode

Las tramas empiezan con el prefijo AT, seguido del comando AT que deseemos emplear y a continuación algunos valores de configuración que dependen del comando en cuestión.

```
// Enter command mode
+++OK

// Read the ID setting
ATID <Enter>
0

// Change the ID setting
ATID 2015 <Enter>
OK
```

Figure 19.- Ejemplo modo AT

También es posible enviar múltiples mensajes AT, separados por una coma al mismo destinatario

2.3.2.4.2 MODO API

Este modo de programación trabaja como una evolución más compleja del modo transparente, que se verá a continuación. Este modo permite mediante el uso de una trama estructurada, donde los datos se colocan en marcos ordenados siempre de la misma forma.

Existen 3 opciones en modo API, la primera AP=0, en este caso el modo API esta desactivado y el funcionamiento de los módulos Xbee es en modo transparente. El segundo caso AP=1, en el que se trata del modo API operation. Y por último el tercer

caso AP=2, que se trata de modo api con “escaped characters”.

Este último modo difiere del modo 1 en el caso en que a medio mensaje se reciba un nuevo parámetro start de trama, en ese caso descarta el mensaje precedente e inicia de nuevo la lectura en ese punto. Se utiliza para evitar mensajes incompletos o mensajes erróneos.
Las tramas de datos siempre empiezan por el mismo parámetro start “7E”, a continuación se incluye el largo del mensaje en bytes, seguidamente se añade el identificador API i el ID del tipo de frame que vamos a utilizar, para seguidamente añadir la dirección del destinatario el paquete de datos que queremos enviar y por último el checksum de verificación de mensaje completo.

Byte(s)	Description
7E	Start delimiter
00 0A	Length bytes
01	API identifier
01	API frame ID
50 01	Destination address low
00	Option byte
48 65 6C 6C 6F	Data packet
B8	Checksum

Figure 20.- Partes de la trama API

Existen varios API frames para diferentes adquisiciones de datos como por ejemplo: el envío de un comando AT, enviar un paquete de datos, enviar un comando AT remoto, leer el estado del modem, recibir un paquete de información, recibir un resumen de la entradas y salidas de un módulo.

2.3.2.5 MODO TRANSPARENTE

Este modo de trabajo es el modo por defecto de los módulos Xbee, no es necesario preconfigurar los módulos ni cambiar los firmwares internos de este.

Su función es la de sustituir un cable serie, Rx y Tx, de manera inalámbrica.

En este modo de trabajo todo lo que entra por el pin 3 del Xbee,Rx , se guarda en el buffer de entrada y luego es transmitido y todo lo que se recibe como paquete se guarda en el buffer de salida y a continuación se envía por el pin 2, o Tx.

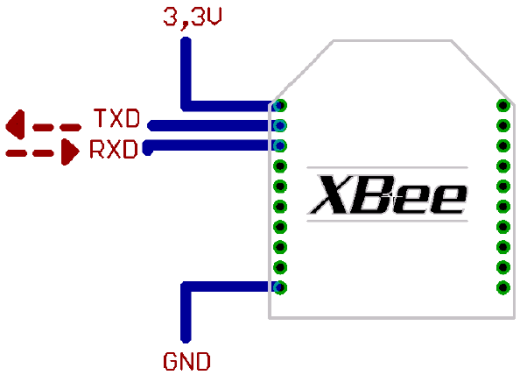


Figure 21.- Conexión modo transparente Xbee

Existe una opción de trabajo mixta en la que se configura el nodo final en modo transparente y el nodo coordinador en modo API de tal forma que podemos enviar datos vía el puerto serie del nodo final y recibirlos en el coordinador en una trama API obteniendo así la información del emisor que ha enviado estos datos.

2.3 ENERGY HARVESTING

Por el termino de “energy harvesting” se entiende la recolección de energía disponible en el entorno, para convertirla en energía útil para nuestra aplicación.
Esta energía puede ser utilizada en aplicación directa o bien se recoge y se almacena en la batería con la finalidad de alargar la vida útil de uno de nuestros módulos IOT.

Esto facilita la colocación de sensores sin tener que cablear ni montar una red de alimentación para ellos, cosa que permite variar sus localizaciones e incluso cambiarlos de maquina o utilizarlos para otra aplicación.

El proceso de captación de energía puede tomar diferentes formas dependiendo de la fuente de la que provenga esta energía. En la configuración más sencilla para poder llevar a cabo de manera correcta la recolección de la energía son necesarios una serie de componentes.

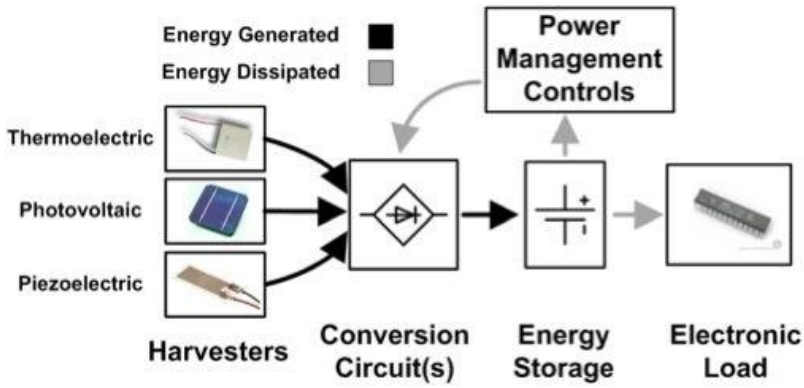


Figure 22.- Componentes básicos de un sistema de recolección de energía

2.3.3 ELEMENTOS DEL SISTEMA DE RECOLECCION DE LA ENERGIA

2.3.3.1 TRANSDUCER O HARVESTERS

La recolección de la energía puede provenir de diferentes fuentes, estos dispositivos son los encargados de convertir la energía del ambiente en energía eléctrica.

Los tipos más típicos de transductores son:

- Energía lumínica vía placas fotovoltaicas
- Energía térmica vía placas termoeléctricas
- Energía magnética vía placas inductivas
- Energía radio frecuencia vía placas RF
- Energía cinética o de vibración vía placas piezoeléctricas

2.3.3.2 POWER MANAGEMENT

Este dispositivo es el encargado de convertir la energía de las fuentes en energía útil para nuestro dispositivo.
Este proceso incluye reguladores de tensión y corriente, además de circuitos de control más complejos que permiten control de potencia y la adaptan a las necesidades de nuestro sistema.

2.3.3.3 ENERGY STORAGE

Los componentes almacenadores de la energía usualmente son baterías o capacitores.

2.3.4 DESCRIPCION DE LAS TECNOLOGIAS DE RECOLECCION DE ENERGIA

En este apartado se incluye un pequeño estudio de las diferentes tecnologías de captación de energía residual que se utilizan.

2.3.4.1 ENERGIA SOLAR

En el caso de la energía solar su recolección se produce mediante la utilización de placas fotovoltaicas, en este caso si nos encontramos en un entorno industrial cerrado, en el que la iluminación es artificial, lo mas usual debido a la poca intensidad de la luz en el ambiente es utilizar la energía para recargar las baterías del dispositivo, en el caso de poder disponer de luz solar se podría considerar la utilización de energía solar para el funcionamiento intrínseco del dispositivo, como se utiliza en algunas calculadoras por ejemplo.

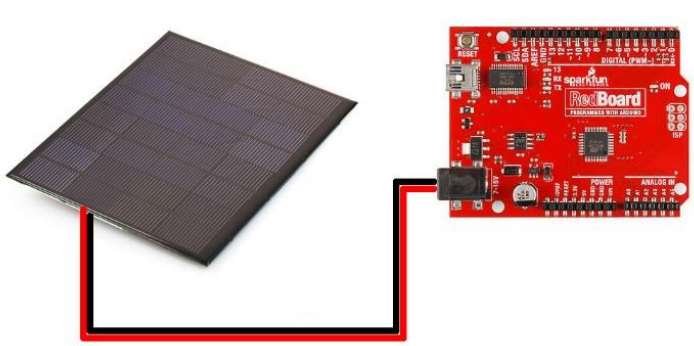


Figure 23.- Placa solar para recolección de energía residual

2.3.4.2 ENERGIA TERMICA

En el caso de la energía térmica su recolección se produce mediante la utilización de placas termoeléctricas estas placas están basadas en el efecto Seebeck por el cual al generar una diferencia de temperatura entre las dos caras de la placa se genera energía eléctrica.

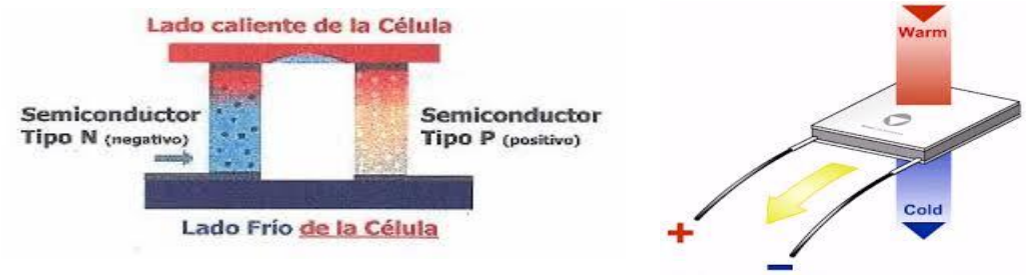


Figure 24.- Efecto fotoeléctrico (Seebeck)

2.3.4.3 ENERGIA RADIO FRECUENCIA

Este tipo de energía se recoge mediante una antena de recolección de energía de RF, y mediante un transductor de radio frecuencia de genera electricidad.

Usualmente el convertidor convierte la señal RF a corriente DC 5.25V con una intensidad de hasta 50mA.

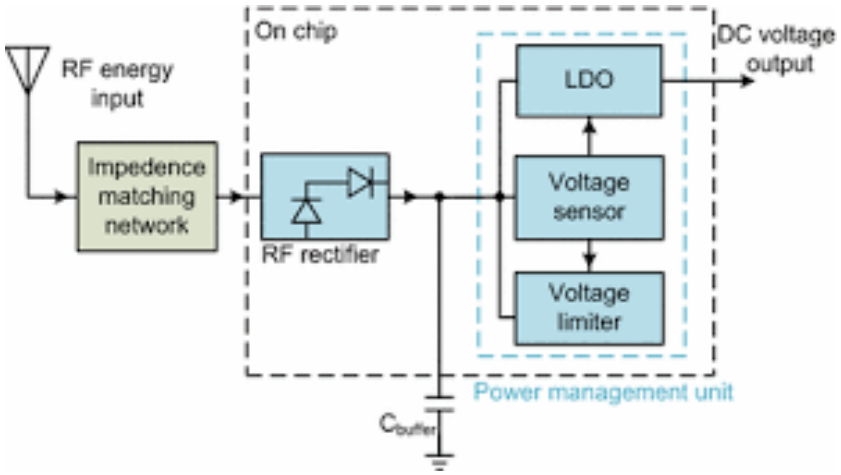


Figure 25.- Energía RF a DC

2.3.4.4 ENERGIA CINETICA O DE VIBRACION

Los transductores piezoeléctricos son los encargados de convertir el movimiento en energía eléctrica.
El transductor convierte la energía cinética a voltaje AC para posteriormente rectificarlo, regularlo y almacenarlo en las baterías.

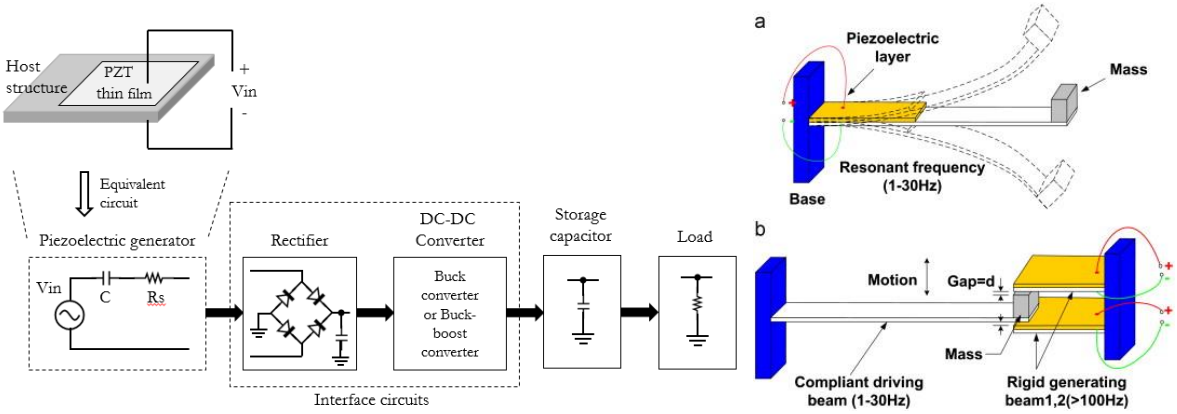


Figure 26.- Efecto piezoeléctrico y conversión a DC

2.3.4.5 COMBINACION DE ENERGIAS

Existen una serie de dispositivos que permiten regular varias fuentes diferentes y obtener una sola salida de voltaje constante para alimentar nuestro sistema IOT.
Un ejemplo de esto es el dispositivo MAX17710, este dispositivo permite controlar fuentes de bajo voltaje como las citadas hasta ahora, fuentes de alto voltaje y baterías.

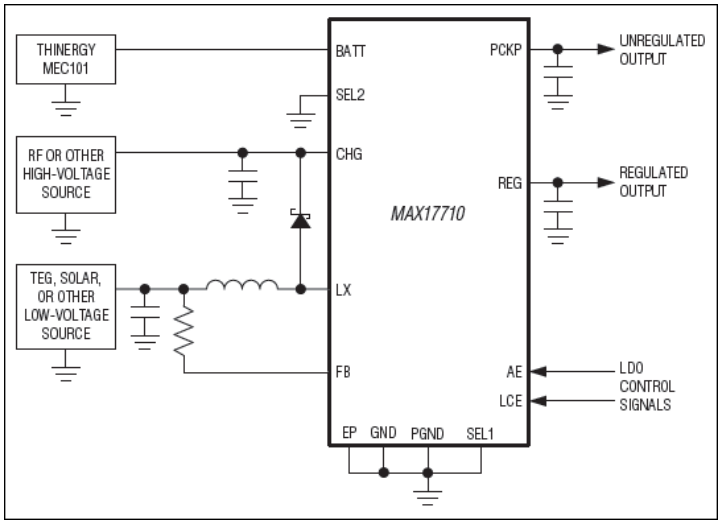


Figure 27.- Regulador MAX17710

3. DISEÑO DE LA RED DE DISPOSITIVOS “LOW POWER”

3.1 DESCRIPCION

La parte práctica de este trabajo consiste en la creación de un dispositivo de bajo consumo que monitorice y controle un proceso industrial.
Este proyecto se corresponde con una primera fase de prototipado y desarrollo dentro de un proyecto más grande de ámbito industrial y aplicación real en mi lugar de trabajo.

En primer lugar, se decide utilizar este proyecto como estudio de las posibilidades y las posibles aplicaciones de una red de monitorización y control de procesos industriales. Además, fruto de una necesidad real del departamento de producción, se decide hacer una primera implementación en el control de la producción de la sala de inyección de plásticos.

Este control debe permitir monitorizar los principales parámetros de las máquinas de manera externa, entre estos parámetros podemos incluir la producción en sí, la temperatura del líquido refrigerante, la temperatura del molde, y otros parámetros intrínsecos de cada tipo de producción. También se debe controlar el correcto funcionamiento de los periféricos que permiten el buen funcionamiento de la maquinaria, en este caso se trata del equipo de frio, el encargado de refrigerar el líquido de refrigeración y el equipo de aire, que es el encargado de suministrar aire comprimido a los accionadores neumáticos que extraen las piezas de los moldes. Por último, se decide incorporar un nodo de monitorización de la temperatura y humedad de la sala, ya que son dos factores determinantes en el buen acabado de las piezas y el correcto funcionamiento de las maquinas.

Esta red de nodos de bajo consumo debe ser de fácil instalación siguiendo el principio “plug and play”, además de ser autoalimentada, en los casos que sea posible, mediante energía residual reutilizada del ambiente industrial en el que se encuentre.

Una vez obtenidos los datos, estos son almacenados y controlados, mediante un controlador PC , para ser utilizados posteriormente como análisis del proceso industrial. Además, la necesidad real de dejar las maquinas sin atención durante largos periodos de horas, únicamente controladas por este sistema, requiere de un programa de toma de decisiones. Por tanto, con los datos obtenidos se desarrolla un programa que, en el caso de que se produzca algún error en la producción se avise al operario en cuestión para que proceda a la solución de este, y en el caso de funcionamiento totalmente autónomo sin trabajadores, turno de noche, proceda a apagar los periféricos para no generar futuros problemas debido al funcionamiento anormal de la instalación.

Las características principales de esta red es su capacidad de autoalimentación mediante “harvesting”, el bajo consumo, la facilidad de instalación, la utilización de los datos obtenidos para de manera automática tomar decisiones sobre el propio proceso de control del buen funcionamiento de este y de minimización de futuros problemas debido al funcionamiento anormal del proceso.

3.3 HARDWARE

En este apartado se incluye el hardware elegido para la construcción de este conjunto de dispositivos.

3.3.1 XBEE

Se ha decidido utilizar comunicaciones de bajo consumo vía radio frecuencia mediante la tecnología del protocolo Zigbee.

Se ha elegido Xbee porque dentro de su segmento permite una mayor capacidad de generación de diferentes tipos de redes, además de mucha variedad de modos de bajo consumo, que permiten adaptar mejor este dispositivo a las diversas aplicaciones posibles, además su bajo coste representa un factor positivo en la decisión.

Entre de los módulos Xbee se ha elegido la serie 1 debido a su facilidad de uso ya que es la primera serie que salió al mercado y es la que está más probada y de la que se encuentra más información al respecto. Se descarta la serie 2 debido a la no compatibilidad con las demás series y por la cantidad de problemas que se pueden leer en otros estudios en su utilización. La serie 3 justo acaba de salir al mercado y no se conoce aún su fiabilidad.

Figure 28.- Xbee S1 Pro

Se ha elegido la variante PRO ya que ésta tiene un alcance mayor, y puesto que la aplicación se llevará a cabo en un entorno interior en el que existen diversidad de obstáculos, se podrá lograr un mayor alcance.

Por otro lado, se considera indispensable que los Xbee dispongan conexión U.F.L para poder conectar la antena en el exterior de la caja nodo. No obstante, para las pruebas del prototipo no se ha considerado que fuera indispensable ya que se ha preferido tener un mayor acceso al nodo y controlar su buen funcionamiento antes de incluir una carcasa.

3.3.2 SENSORES

La utilización de los sensores adecuados para cada aplicación es un factor determinante. En esta aplicación se ha probado diferentes sensores para tomar datos parecidos con la finalidad de poder conocer sus características y de cara a futuras aplicaciones tener un mayor conocimiento de sensores diferentes. Los sensores utilizados en este prototipo son:

3.3.2.1 SENSOR DE TEMPERATURA Y HUMEDAD AM2302

El AM2302 es un sensor digital de temperatura y humedad que incorpora una carcasa de protección. Para medir la humedad utiliza un sensor de humedad capacitivo y para medir la temperatura emplea un termistor.
La única limitación de este sensor es que solo permite hacer una lectura cada 2 segundos.

Las especificaciones de este sensor son las siguientes:

- Voltaje: 3-5 V DC
- Corriente máxima: 2.5 mA
- Precisión humedad 2-5%
- Rango de temperatura: -40ºC a 80ºC
- Precisión temperatura: ± 0.5ºC
- Dimensiones: 27mm x 59mm x 13.5mm
- VCC: cable rojo
- Data out: cable amarillo
- Ground: cable negro

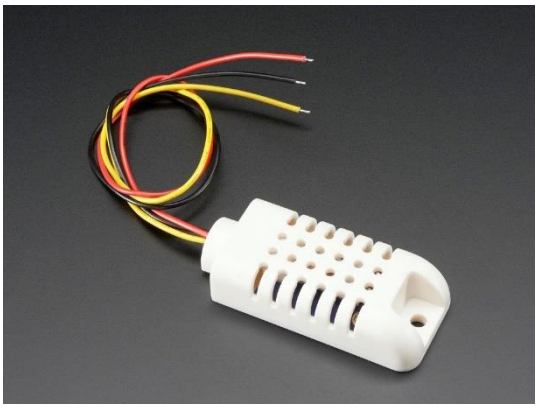


Figure 29.-Sensor AM2302

3.3.2.2 SENSOR DE TEMPERATURA ME DS18B20

Este sensor es un sensor de temperatura impermeable. Es un sensor adecuado para la detección de temperatura inmersiva o en contacto.
El sensor DS18B20 es un sensor digital que trabaja por un solo cable, proporciona lecturas de 9 a 12 bits. Tiene unas dimensiones de 6mm de diámetro y unos 50 mm de largo.
Sus especificaciones eléctricas son:

- Tensión de alimentación: 3 – 5.5 V DC
- Rango de temperaturas: -55ºC a 100ºC

- Precisión: ± 0.5ºC

Figure 30.- Sensor ME DS18B20

3.3.3 BATERIAS

En el caso de las baterías, como el objetivo era ver las distintas opciones, se ha decidido probar la opción no recargable de vida limitada, aunque como veremos posteriormente en el cálculo del consumo de la aplicación, de vida bastante larga, y la opción recargable mediante métodos de recolección de energía sobrante del entorno.

3.3.3.1 BATERIAS NO RECARGABLES

En el caso de la batería no recargable y contando que el voltaje de alimentación del arduino mini pro y de los xbee es de 3.3v. Se decide implementar una batería de Litio de alta capacidad de 3.6V. El modelo ER34615M de tipo D.

Figure 31.- Pila industrial ER34615M

Este tipo de pilas industriales incorporan una salida estable, un protector por sobre voltaje de 5A y una salida regulada de corriente de hasta 2A.

La capacidad de la pila es de 14.5Ah, que para la aplicación, como veremos en el cálculo del consume, es más que suficiente para una vida útil superior a los 2 años.

En la siguiente imagen podemos observar la curva de descarga, así como la relación entre capacidad y corriente a diferentes temperaturas.

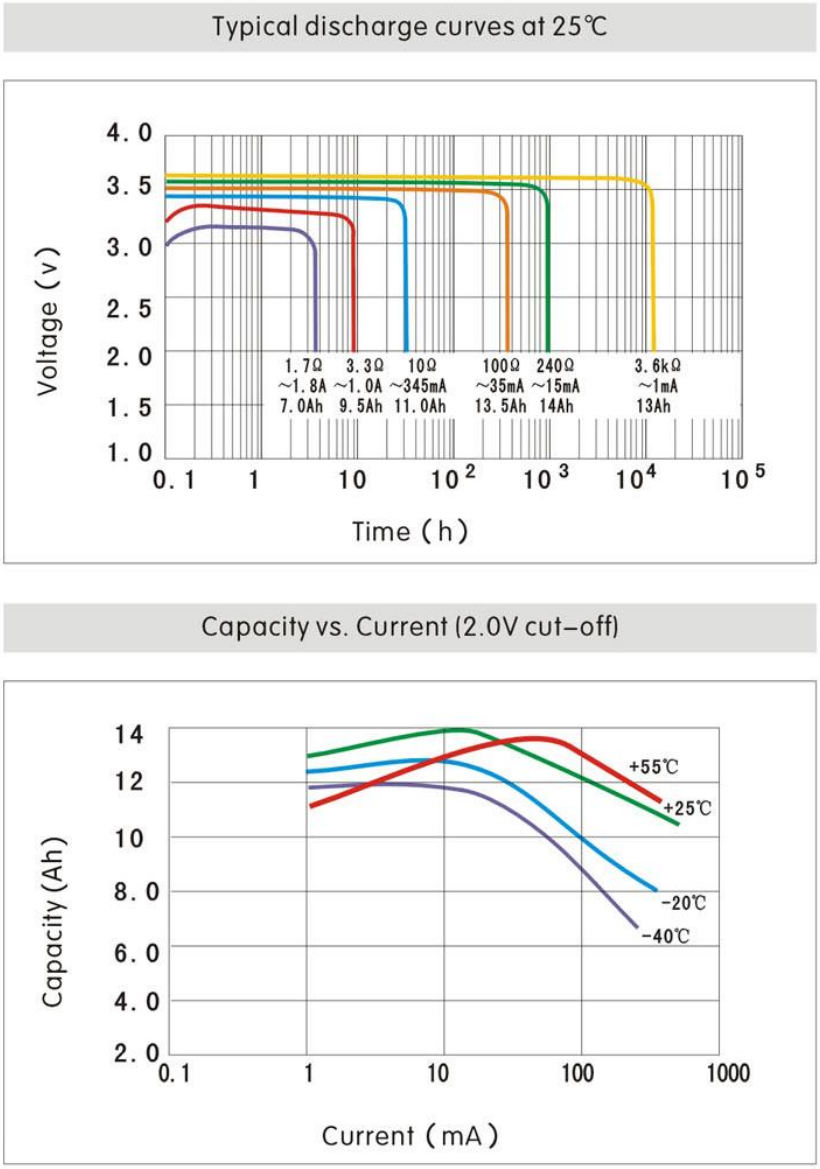


Figure 32.- Curvas características pila ER34615M

***para más información ver anexo 9.4**

3.3.4 CONTROLADOR

En esta aplicación se ha utilizado 4 controladores diferentes para analizar las posibilidades de cada uno de ellos.

3.3.4.1 ARDUINO UNO

En el caso del arduino UNO, se trata de un controlador basado en el microprocesador ATMEL ATmega 328. Es considerado el procesador Arduino más básico, y por ello tiene algunas limitaciones como es el caso de que solo incorpora un puerto serie físico.

Es cierto que existe la posibilidad de generar puertos serie vía software, pero estos puertos no funcionan igual de bien que los puertos vía hardware ya que no incorporan un buffer físico de entrada.

En la aplicación el arduino UNO se utiliza en las primeras pruebas como controlador de la aplicación, no obstante llega un momento en que sus limitaciones obligan a mejorar de procesador y se opta por cambiar al arduino mega. Como se ha comentado antes, la imposibilidad de trabajar con dos puertos serie de manera fluida simultáneamente para conectar la recepción de datos vía XBee y el pc como registrador de datos, motiva el cambio a una versión más completa de procesador.

Figure 33.- Arduino UNO

Aunque para esta aplicación se requieren diversos puertos serie en el nodo coordinador, haber estudiado las posibilidades del arduino UNO, ha permitido conocer sus limitaciones de cara a otras implementaciones más sencillas o con unos requerimientos distintos.
Su bajo coste y su sencilla programación permiten una implementación muy ágil, además existe una gran cantidad de recursos vía web con información.

***para más información ver anexo 9.1**

3.3.4.2 ARDUINO MINI PRO

En el caso del Arduino mini pro, se utiliza en los mismos nodos como procesador de datos de diferentes sensores en el caso de tener que acondicionar datos o realizar alguna operación previa al envío de estos datos. Además, te permite generar una sola cadena de datos más fácil de interpretar desde el receptor con los diferentes datos ya acondicionados de los sensores.

Utilizar un arduino en cada nodo permite también tomar decisiones en la recepción de cadenas de datos, e implementar acciones más complejas, que con la utilización únicamente del Xbee.
Además, este arduino al estar alimentado también a 3.3V y tener un consumo muy bajo además de incorporar la librería “low power” de Arduino lo convierte en el procesador idóneo para esta aplicación.

Otro factor diferenciador de este arduino es la frecuencia de trabajo del microprocesador, ésta es de sólo 8MHz, es decir, la mitad de la frecuencia de los demás Arduinos. Esto genera un menor consumo de corriente que es favorable para aplicaciones de iot en tiempo discreto.

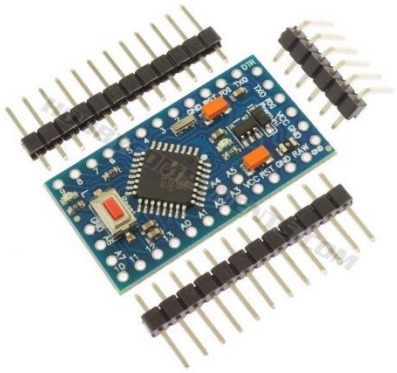


Figure 34.- Arduino MINI PRO

***para ver el datasheet del dispositivo ver el anexo 9.2**

3.3.4.3 ARDUINO MEGA

La utilización del Arduino Mega como procesador sustituye arduino UNO, como ventaja destacar sus 4 puertos serie en hardware.

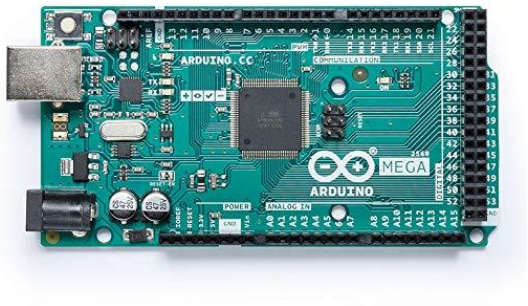


Figure 35.-Arduino MEGA

***para ver el datasheet del dispositivo ver el anexo 9.3**

3.3.4.4 PC INDUSTRIAL

Se ha utilizado un pc industrial como receptor de los datos y para mediante el software matemático Matlab, el cálculo de las estadísticas del proceso, así como la toma de decisiones respecto el correcto funcionamiento del proceso industrial.

Figure 36.- PC industrial

3.3. SOFTWARE

En este apartado podemos encontrar el software elegido para su utilización en la aplicación.

3.4.1 XCTU

XCTU, también conocido como “Xbee Configuration and Test Utility” es el programa encargado de la configuración de los módulos xbee. Este programa viene dado por la propia empresa Digi y nos permite desde variar cualquier parámetro de configuración hasta la variación del firmware interno para poder acceder a distintas configuraciones. También incorpora opciones para comprobar el correcto funcionamiento de nuestra red.

Este software, mediante una interfaz gráfica, permite variar parámetros de los nodos, dependiendo del modo en el que trabajemos, incluso en nodos conectados a la red que no tienen porque estar conectados al ordenador vía cable, es decir, desde un nodo coordinador se pueden variar parámetros de configuración de los nodos red a distancia. En la imagen podemos ver el primero de los tres menús de la aplicación, en este caso

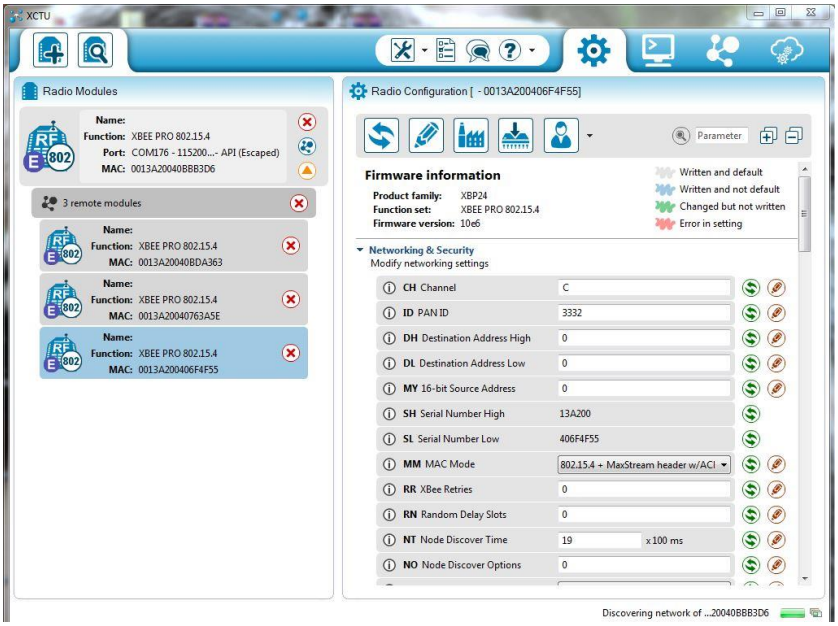


Figure 37.- XCTU software, pestaña configuración

simbolizado por un engranaje. Este menú se puede variar la configuración de los nodos xbee. Para destacar alguna característica podemos ver la dirección de destino a la que este nodo va a enviar sus mensajes, DH y DL.

A continuación, en la siguiente imagen podemos observar el segundo de los menús de CTU. En este caso se corresponde al menú que incorpora las opciones de test del xbee. En esta imagen podemos observar la llegada de tramas en modo AT. En el lado izquierdo el texto ASCII i en el derecho el mensaje en hexadecimal.

También incorpora un generador asistido de paquetes de datos para facilitar el envío desde el mismo pc a otros nodos de la red.

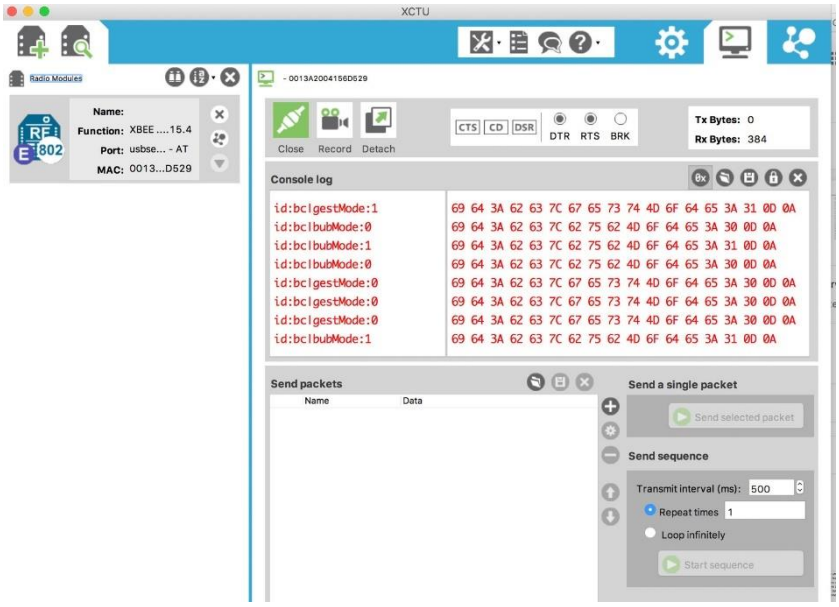


Figure 38.- Menú test XCTU

Por último, el menú de más a la derecha nos muestra nuestra configuración red y los nodos que están conectados entre sí.

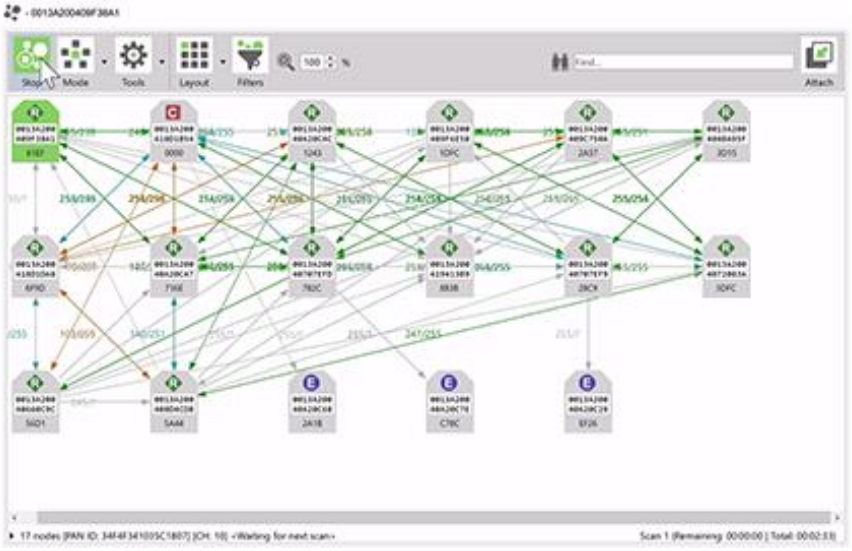


Figure 39.- Conexiones red entre nodos XBEE

3.4.2 ADECUADOR DE DATOS

Para adecuar los datos en los nodos y del conjunto se utilizan arduinos programados mediante su propio software Arduino ide.
Mediante este programa se adecuan en los nodos finales los datos de los sensores y se mandan por el puerto serie.
En el nodo coordinador, que se encarga de recibir todos los datos por el puerto serie y enviarlos al ordenador mediante otro puerto serie, los acaba de adecuar leyendo las tramas api de arduino y mediante su descodificación y control del checksum verificando quien envía, que longitud tiene el mensaje y que tipo de mensaje es, y envía estas tramas al ordenador.

3.4.3 CONTROLADOR DE DATOS

Como controlador de datos se ha decidido utilizar Matlab como herramienta de cálculo matemática. En este caso la decisión de basa en que ya se posee cierta experiencia en la utilización de este software, no obstante, se ha estudiado la posibilidad de utilizar herramientas de big data como AZURE o las herramientas de Google. En fases más avanzadas del proyecto se testarán estas otras opciones.

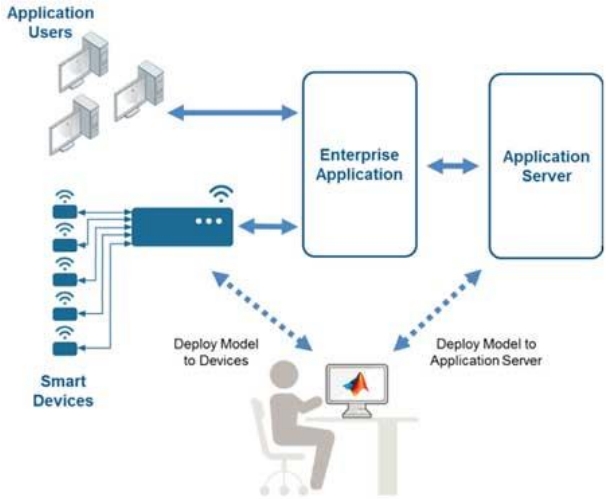


Figure 40.- Utilización de Matlab como herramienta de control

Matlab es un software matemático que permite la manipulación de datos de manera ágil y su representación mediante gráficos. Además, dispone de una herramienta grafica llamada Simulink, que permite generar un entorno más visual a la aplicación.

En Matlab se leen las tramas que provienen del adecuador y se realizan acciones dependiendo de quién envíe esos datos.

Con la finalidad de generar un registro todos los datos tratados en Matlab serán exportados a un fichero de Excel y almacenados en este incorporando la fecha y hora en que han sido tomados.

También se crea una aplicación tipo interface mediante el generador de aplicaciones de Matlab que nos permitirá tener un entorno grafico para manipular la aplicación.

3.4. LOW POWER

En aplicaciones de monitorización dentro de lo que se considera el internet de las cosas, en las que los módulos son usualmente alimentados con baterías al no ser necesaria su monitorización en tiempo continuo, es muy importante que los dispositivos tengan el consumo eléctrico menor posible.

Esto es lo que se conoce como electrónica de bajo consumo “low power”.
En esta aplicación se puede incidir en varios aspectos para disminuir el consumo. Por un lado, los módulos xbee disponen de varios modos de “sueño” en los que se puede dejar el dispositivo con un consumo muy muy bajo mientras no es necesario enviar datos.

En segundo lugar, el software de arduino dispone de una librería, LowPower.h, que permite dormir la placa y despertarla por ejemplo por una interrupción en un pin.

De este modo controlando el periodo de muestreo desde las opciones de sueño de xbee, es posible que al despertarse ponga en modo alto, “HIGH”, uno de sus pines y utilizar este para alimentar el arduino y los sensores. De tal forma que tomamos un dato y lo mandamos controlando el proceso desde el módulo Xbee que en el momento que termine de mandar procederá a volver al estado de sueño y por tanto apagara la alimentación a los sensores y la alimentación a arduino.

Por último, la elección de arduinos a 3.3V para los nodos finales y con una frecuencia de procesador menor, nos garantizan un menor consumo que a su vez implementando una serie de pequeñas modificaciones en la placa a nivel hardware, nos permitirá disminuir todavía más el consumo de esta eliminando ciertos elementos complementarios que solo nos generan perdidas de energía y no realizan ninguna tarea vital en el funcionamiento de la placa arduino.

3.5.1 IMPLEMENTACION LOW POWER XBEE

En el caso de las redes de nodos tipo red,”mesh”, existe aparte de los modos de sueño especificados de cualquier nodo xbee u modo de sueño síncrono.

Este modo de sueño permite sincronizar el periodo de sueño de todos los dispositivos de la red. Formando un ciclo en que por un tiempo todos los nodos están apagados y pasado ese tiempo los nodos se activan un periodo especificado hasta su vuelta al estado de sueño.

Para hacer funcionar este modo de sueño es necesario tener un dispositivo que se encarga del control del sueño de la red y lo coordina. Este dispositivo en modo de sueño 7, SM7, se mantiene despierto todo el tiempo, pero esta sincronizado a la red de sueño. Se encarga de enviar los parámetros de sueño a los nuevos dispositivos que se quieran incorporar a la red, para sincronizar su sueño con los demás.

Es importante tener en cuenta que únicamente habrá comunicación entre los nodos cuando estos estén despiertos.

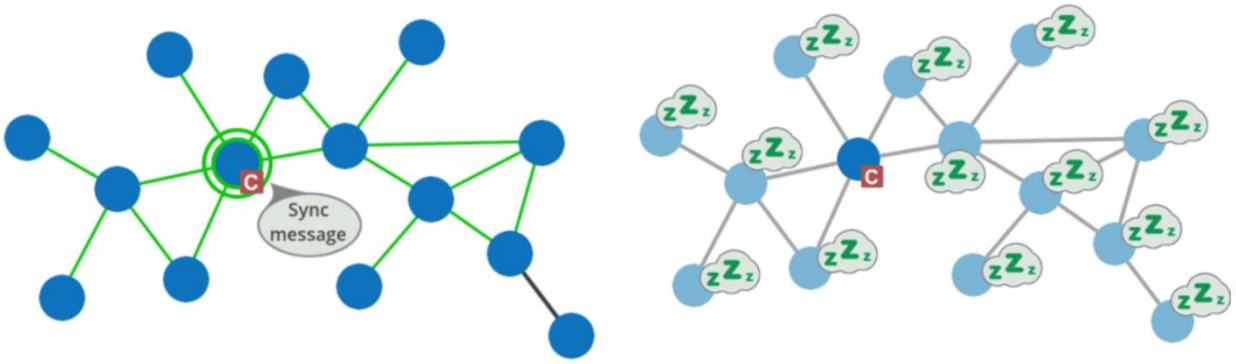


Figure 41.- Sincronización de sueño Xbee

Los demás nodos de la red se encuentran en modo de sueño 8, SM8, estos nodos duermen un tiempo especificado y se despierten simultáneamente con los demás nodos de la red. Intercambian datos entre ellos y regresan al modo de sueño.

En modo “dormido”, no pueden recibir datos ya sea vía serial como inalámbrica.

3.5.1.1 CONFIGURACION MODULOS XBEE

En la configuración como se ha especificado anteriormente es necesario diferenciar entre la configuración de un nodo coordinador de sueño y un nodo final.

En todos los casos la red deberá ser la misma y todos se tendrán que direccionar al router y este al coordinador mediante las direcciones de 64 bits de los dispositivos.

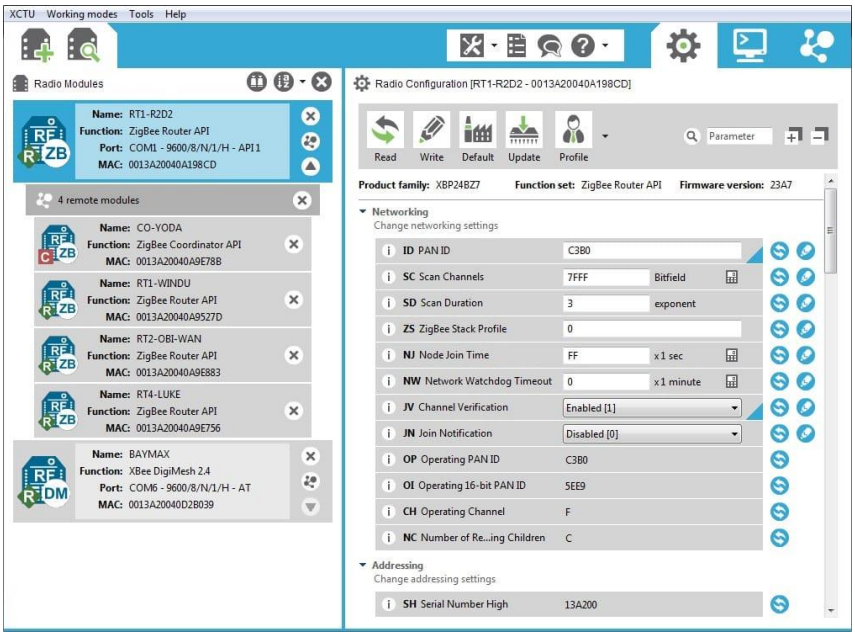


Figure 42.- Xctu configuración id de la red

3.5.1.1.1 CONFIGURACION COORDINADOR

En le caso del nodo coordinador no se duerme nunca pero si que esta coordinado con la red de sueño de manera que recibe los datos y realiza los envion cuando los demás nodos están despiertos.

A continuación se incluye la configuración en XCTU del nodo coordinador.

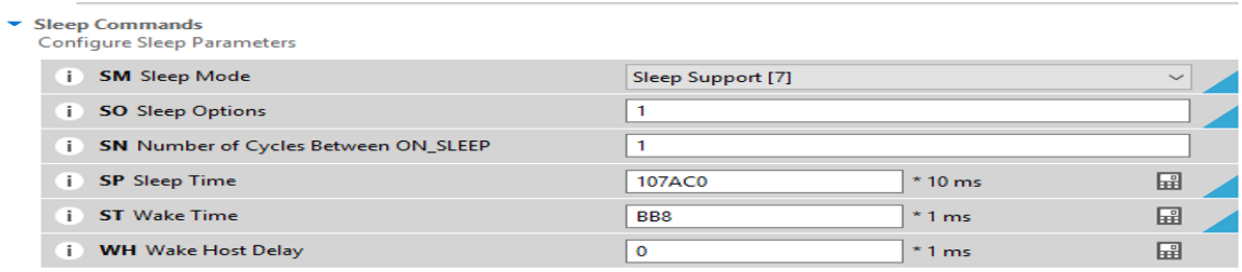


Figure 43.-Sleep comands coordinador

Esta metodología tiene la gran ventaja que controla el sueño de toda la red de sensores, no obstante, si conocemos los periodos exactos de cada dispositivo conseguiremos un consumo todavía inferior ajustando el tiempo del ciclo de sueño en cada uno de los dispositivos internamente.

3.5.1.1.2 CONFIGURACION NODOS

En el caso de los nodos es importante guardar un pin en modo “sleep output”, por el que se alimentará todo el nodo final cuando el xbee se despierte. En este caso el pin 9.

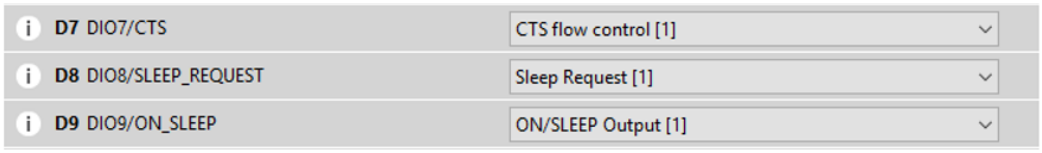


Figure 44.- Pin Sleep Output xbee

3.5.2 IMPLEMENTACION LOW POWER ARDUINO

Respecto a la implementación de “low power” en arduino, existe una librería que via software disminuye el consumo de los dispositivos y permite dormir los dispositivos, la librería lowPower.h, además existen otras librerías creadas por usuarios que complementan la librería de arduino con algunas funcionalidades.

En segundo lugar, existe la posibilidad de disminuir el consumo todavía mas retirando elementos del hardware de la placa que realizan funciones que no son necesarias para el funcionamiento de la placa.
A continuación, se entra más en detalle de la disminución de consumo que representa la aplicación de estos principios.

3.5.2.1 LOW POWER EN ARDUINO MINI PRO

Respecto a los arduino mini pro, son una seria de arduinos con una versión alimentada a 3.3V mas adecuada para aplicaciones de bajo consumo de adquisición discreta de datos, ya que incorporan un procesador mas lento de 8MHz.

En la aplicación los nodos finales incorporan arduinos mini pro que controlan y procesan la adquisición de datos para enviarlas vía Xbee al controlador del proceso.
Estos mediante la utilización de la rutina “sleep_mode_pwr_down”, del microprocesador, nos permite dormir el dispositivo y despertarlo, por ejemplo mediante una interrupción en uno de sus pines.

Además, existen unas modificaciones hardware que disminuyen el consumo de este dispositivo. Estas modificaciones son:

- Desactivar el Led de control de marcha, este led se enciende siempre que el dispositivo este alimentado.

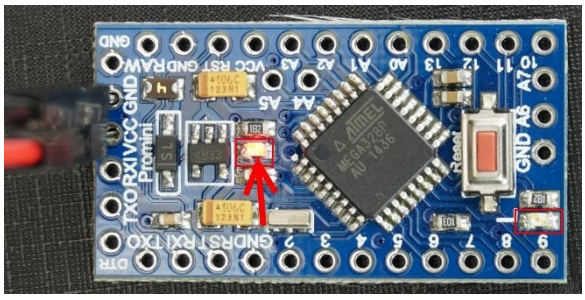


Figure 45.- Desactivar led power arduino mini pro

- Extraer el regulador de voltaje, ya que, a corrientes muy bajas, del orden de 0.1mA, su eficiencia es de solo un 20%. Por otro lado, esto nos obliga a ser más precisos en el voltaje de entrada aunque el propio microprocesador tiene una tolerancia de entre 2.2V a 5.5V

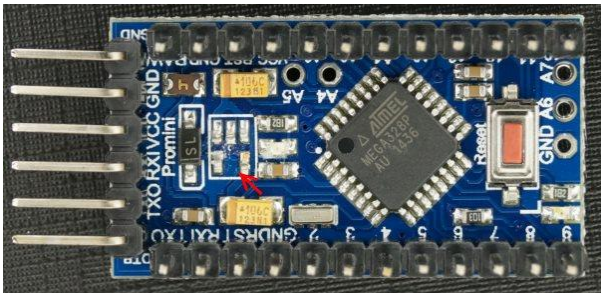


Figure 46.-Extracción Regulador voltaje arduino mini pro

A continuación, podemos observar la disminución del voltaje con las diferentes implementaciones de mejoras para el consumo, tanto en arduino mini pro de 5V y de 3.3V.

ATmega328P Pro Mini Version	PWR Source	State	5.0 V @ 16 MHz	3.3 V @ 8 MHz
Unmodified	RAW Pin	ACT	19.9 mA	4.74 mA
Unmodified	RAW Pin	PDS	3.14 mA	0.90 mA
No Power LED	RAW Pin	ACT	16.9 mA	3.90 mA
No Power LED	RAW Pin	PDS	0.0232 mA*	0.0541 mA*
No Power LED, no Regulator	VCC Pin	ACT	12.7 mA	3.58 mA
No Power LED, no Regulator	VCC Pin	PDS	0.0058 mA	0.0045 mA

Figure 47.-Diminución consumo Arduino Mini Pro

El primer caso se trata del arduino sin modificaciones, el segundo del arduino con la implementación de una librería de bajo consumo vía software, el tercero y cuarto se corresponde con el consumo una vez se ha extraído el led en la opción con software y sin control por software y por último los dos últimos campos de la tabla se corresponden una vez aplicadas las dos posibles mejoras de hardware.

3.5.2.2 LOW POWER ARDUINOS

En el caso de dormir los demás arduinos mediante la librería se aplicarán acciones de disminución del consumo, no obstante, la regulación la controlara xbee y su “red de sueño”. Por tanto, solo se alimentará arduino en el momento de tomar datos ya que se utilizará el pin de xbee que se pone en estado alto para controlar la alimentación del arduino y de los sensores.

La librería lopower.h incorpora la función lowPower. powerDown(), que nos permite dormir para siempre el arduino hasta que reciba un pulso por el pin 13.

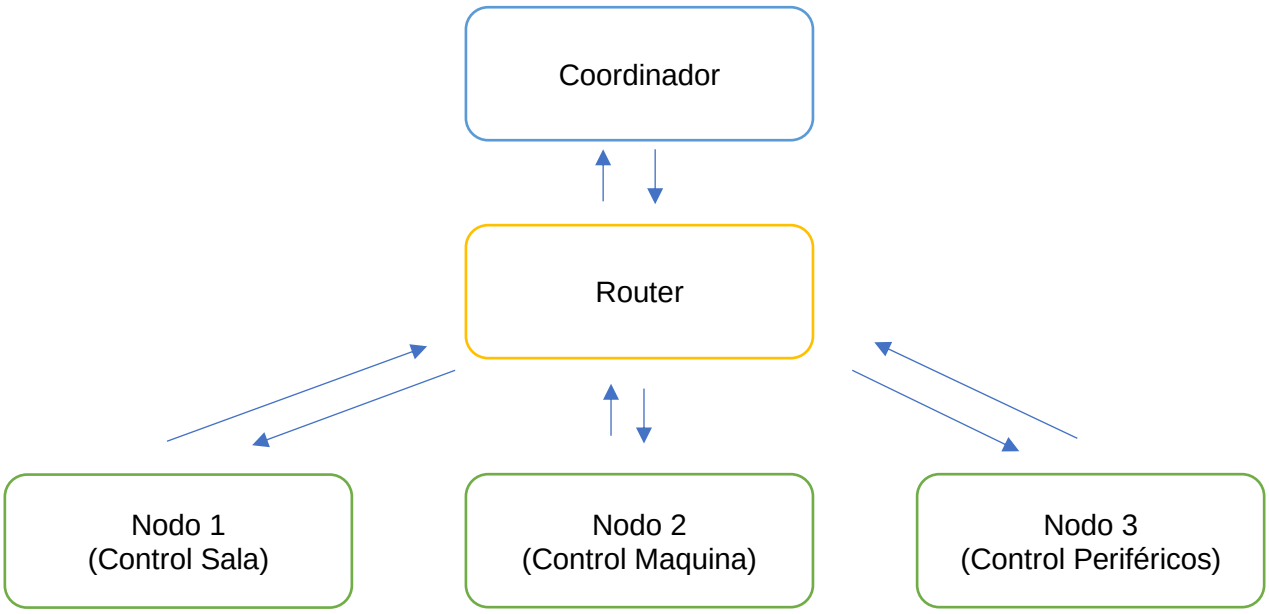
```
#include "LowPower.h"

void setup()
{
    // No setup is required for this library
}

void loop()
{
    // Sleep for 8 s with ADC module and BOD module off
    LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
    // Do something here
    // Example: read sensor, log data, transmit data
}
```

Figure 48.- Función powerDown arduino

4. IMPLEMENTACION DE LA RED DE DISPOSITIVOS “LOW POWER”

En este apartado se incluye la implementación real del prototipo y sus configuraciones. Así como su arquitectura, el diagrama del proceso y la situación real de los nodos en el entorno.

4.1. DIAGRAMA DE BLOQUES



En el diagrama de bloque del prototipo podemos observar que disponemos de 5 nodos interconectados, 3 de ellos son nodos finales, 1 coordinador y 1 router.

Este prototipo intenta simular los requerimientos del sistema real en cuanto a los procesos a nivel de hardware y software. Y ser la muestra en la implementación del proyecto final.

No obstante, que sea un prototipo no implica una disminución de la complejidad ya que se ha intentado realizar y probar todas las operaciones necesarias.

En este prototipo obtendremos en una interfaz manipulable la visualización de los parámetros de sala y maquina y mediante la activación de un parámetro regulador de una alarma accionar un periférico que actúe al respecto.

4.2. ARQUITECTURA DE LA RED DE SENSORES

4.2.1 DISPOSICION DE LOS MODULOS

En el caso del nodo de sala, este nos permite monitorizar la temperatura y la humedad de ésta. Con estos datos podemos controlar que las máquinas trabajen en su funcionamiento óptimo. Los factores de sala a no ser que sean extremos, no impiden el correcto funcionamiento de las maquinas, no obstante, tienen una gran incidencia en el acabado de las piezas ya que el tiempo de “curado” de la pieza dentro del molde y la presión de inyección de plásticos en relación al tiempo de secado del plástico y la temperatura del plástico inyectado se ven influenciadas por la temperatura de la sala.

La humedad, mediante condensación, también puede ocasionar problemas de acabado y brillo de las piezas, así como la aparición de manchas en la superficie de estas, debidas a los gases por presión del plástico evaporando el agua dentro del molde.

Figure 49.-Sala de inyección de plásticos

Respecto al nodo de máquina, este, en el caso del prototipo sólo es un sensor que mide la temperatura del líquido refrigerante del molde, y por tanto, la temperatura a la que está el molde. Este dato es necesario para saber si los equipos de refrigeración están trabajando correctamente. Una temperatura excesiva en el molde nos produciría fallos de pieza, es decir, piezas incompletas o con exceso de plástico.

En el prototipo establecemos un baremo de temperatura y activamos una alarma en el caso de que la monitorización sobrepase este valor. Además de activar, o reactivar, el periférico correspondiente.

Figure 50.- Máquina de inyección de plásticos

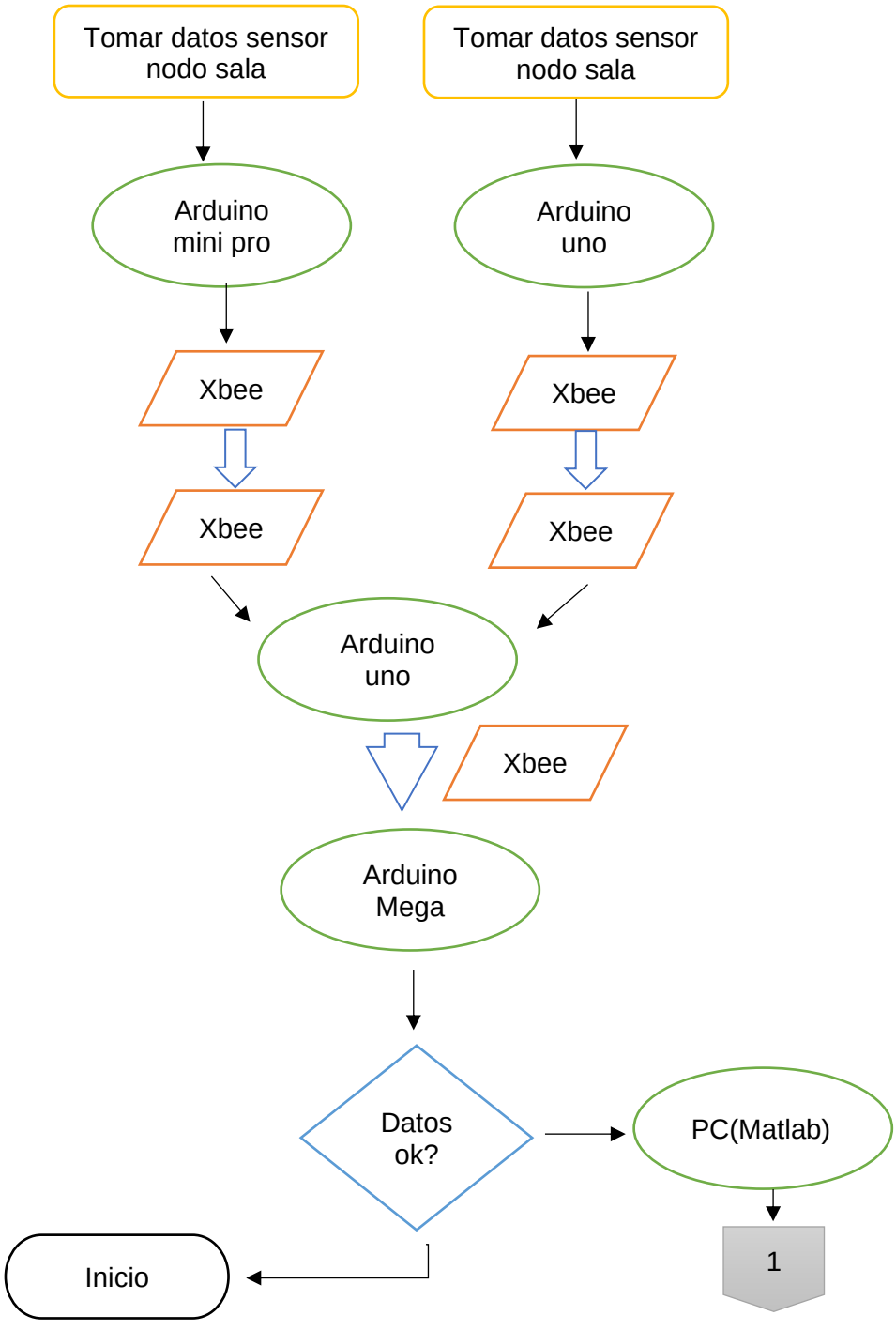
El tercer nodo es el encargado de reactivar los equipos de frio en caso de paro. En el proyecto final además se controlará que, si no se han encendido al enviar la acción, se parará la producción y se alertará vía mensaje y por teléfono al operario de sala para que acuda a solucionar el problema.

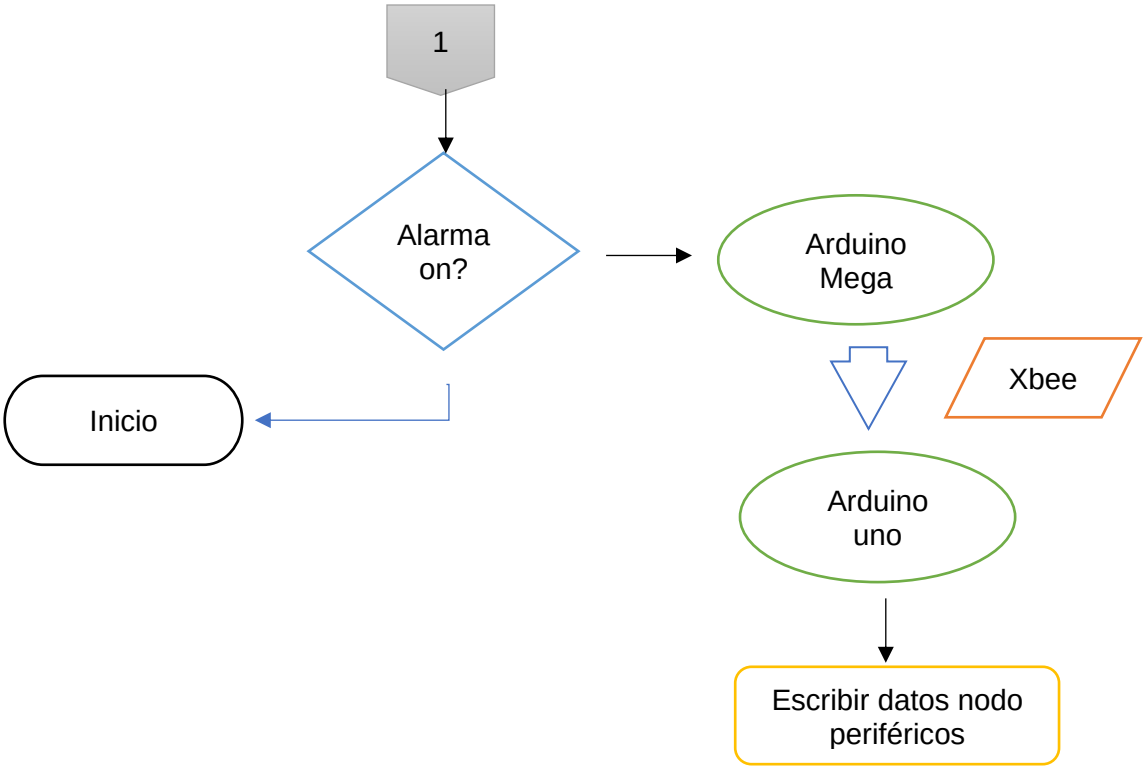
En el proyecto final tenemos otro periférico con un funcionamiento homólogo a este, es el caso del compresor de aire.

Además de los nodos el dispositivo coordinador es un arduino mega, conectado a un xbee por puerto serie, y el dispositivo Router es otro xbee conectado a este mismo arduino.

En el caso de la implementación real, el dispositivo router se encontrará a media distancia entre la sala y el dispositivo coordinador.

4.3. ESQUEMAS DE LOS NODOS





4.3.1 NODO COORDINADOR

El nodo coordinador en nuestro caso es un arduino mega conectado a un xbee como muestra la siguiente figura.

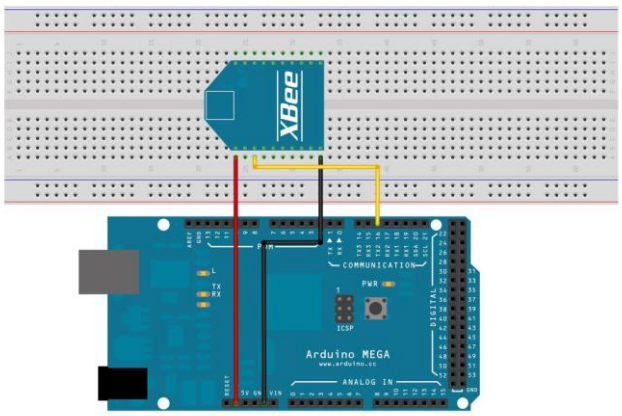


Figure 51.-Conexión Xbee a arduino MEGA

En este caso este nodo además de coordinar la red y recibir toda la información es el encargado de coordinar el sueño y el microprocesador de arduino es el encargado de enviar las tramas de información que lleguen completas una vez verificado el checksum con la estructura correcta para que Matlab las pueda interpretar correctamente.

4.3.2 NODO ROUTER

En el caso del nodo router se compone únicamente de un nodo xbee encargado de recibir los mensajes de los diferentes nodos y enviarlos al coordinador.
Su conexión se realiza de la siguiente manera.

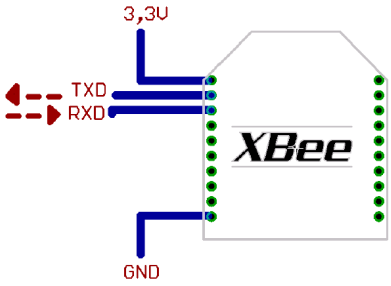


Figure 52.- Alimentación Xbee

4.3.3 NODO CONTROL SALA

El nodo de control de sala se compone por un arduino uno conectado a un sensor por el pin 2 y conectado a un xbee mediante el puerto serie, los pines Rx y Tx.

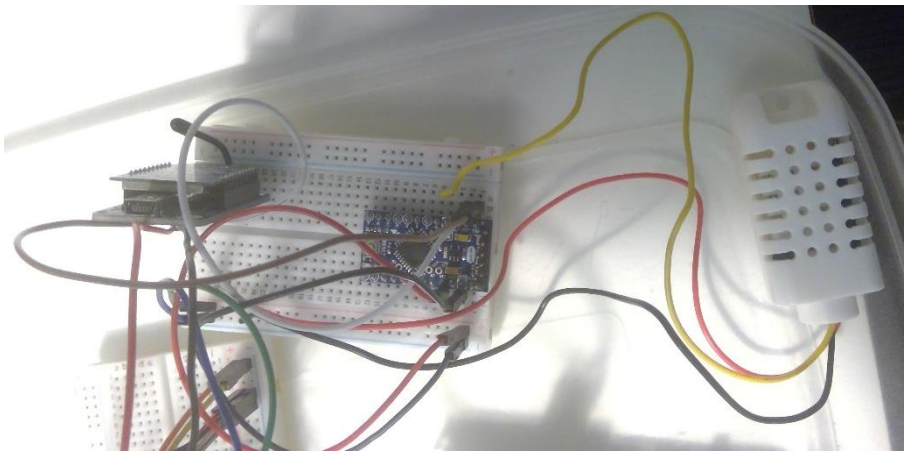


Figure 53.- Nodo Sala

En este caso el sensor Asair, toma datos de temperatura y humedad. Estos datos son recogidos por el arduino mini pro y se reenvían por el puerto serie al Xbee. Este los enviará al nodo router.

4.3.4 NODO CONTROL MAQUINA

En el caso del nodo de control de maquina este compuesto por un sensor ip68, capaz de resistir el agua, que toma datos de temperatura cada 5 segundos y los manda a un arduino UNO, este los renviara via puerto serie al Xbee como en el caso del nodo anterior.

Además, es necesaria la incorporación de una resistencia de un valor superior a 10kOhms como referencia del 0V de tal forma que siempre tiene referencia 0 y por tanto no generara falsas lecturas ni posibles errores derivados de la falta de este componente.

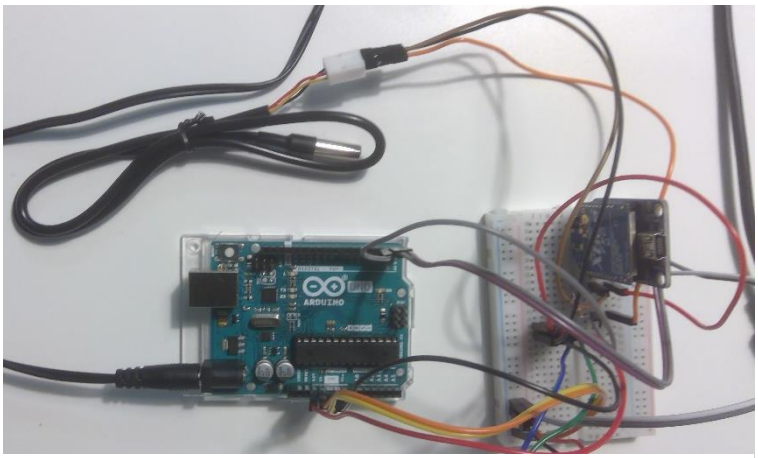


Figure 54.- Nodo control máquina

4.3.5 NODO CONTROL PERIFERICOS

Este nodo tiene una funcionalidad un poco distinta de los demás, este modo trabaja como receptor, es decir, su función principal es activar o reactivar, en el caso de un paro por algún motivo, el funcionamiento de periféricos vitales para el correcto funcionamiento de las máquinas de inyección.

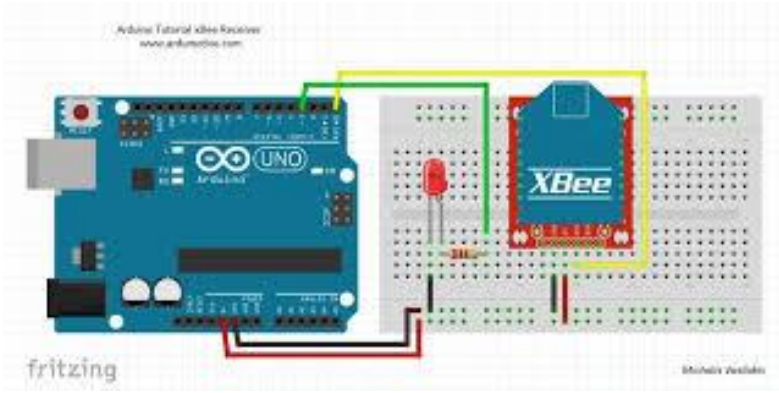
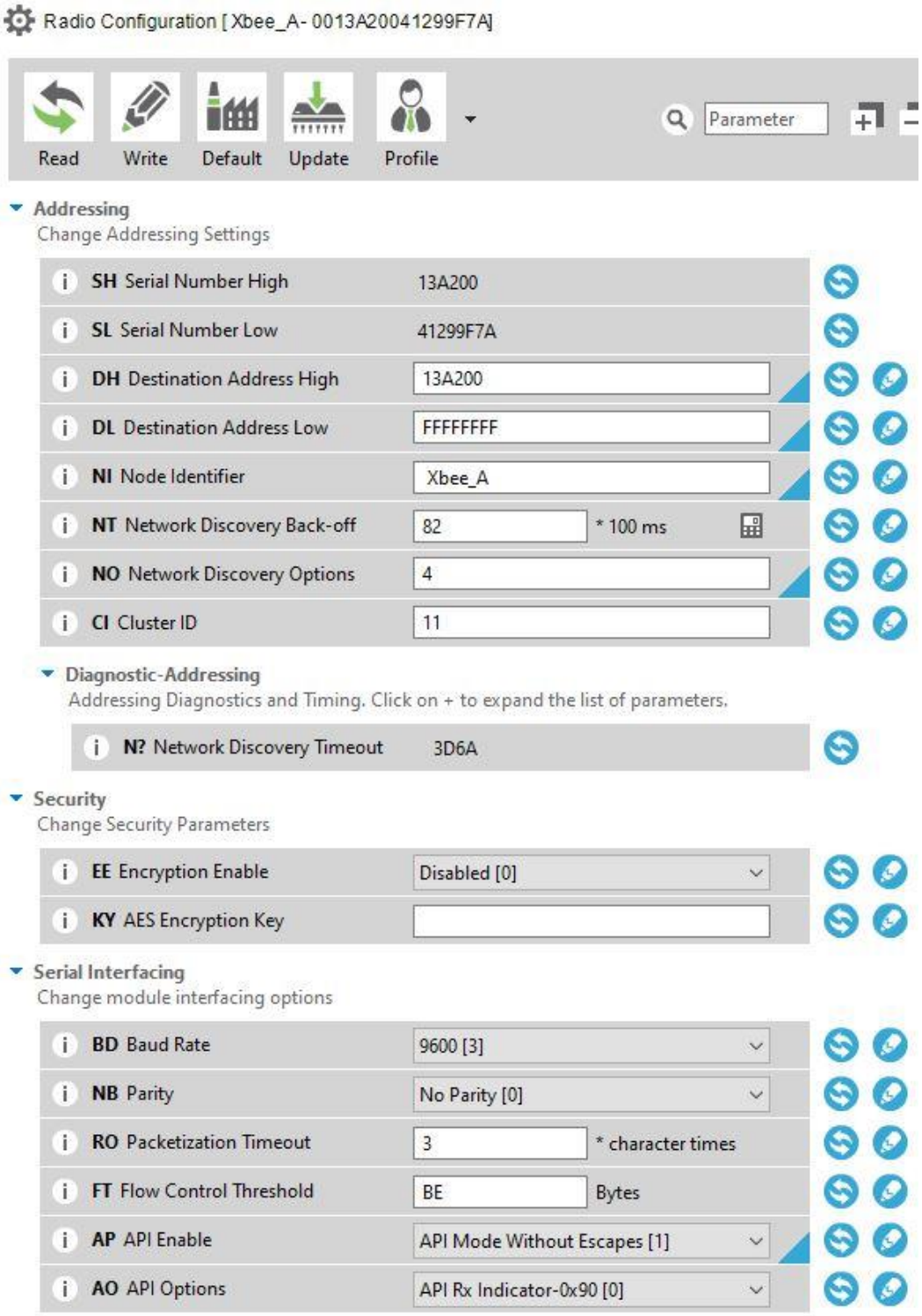


Figure 55.- Nodo control periféricos

4.4. SOFTWARE NODOS

4.4.1 SOFTWARE XBEE

A continuación, se muestra la configuración de los nodos Xbee.



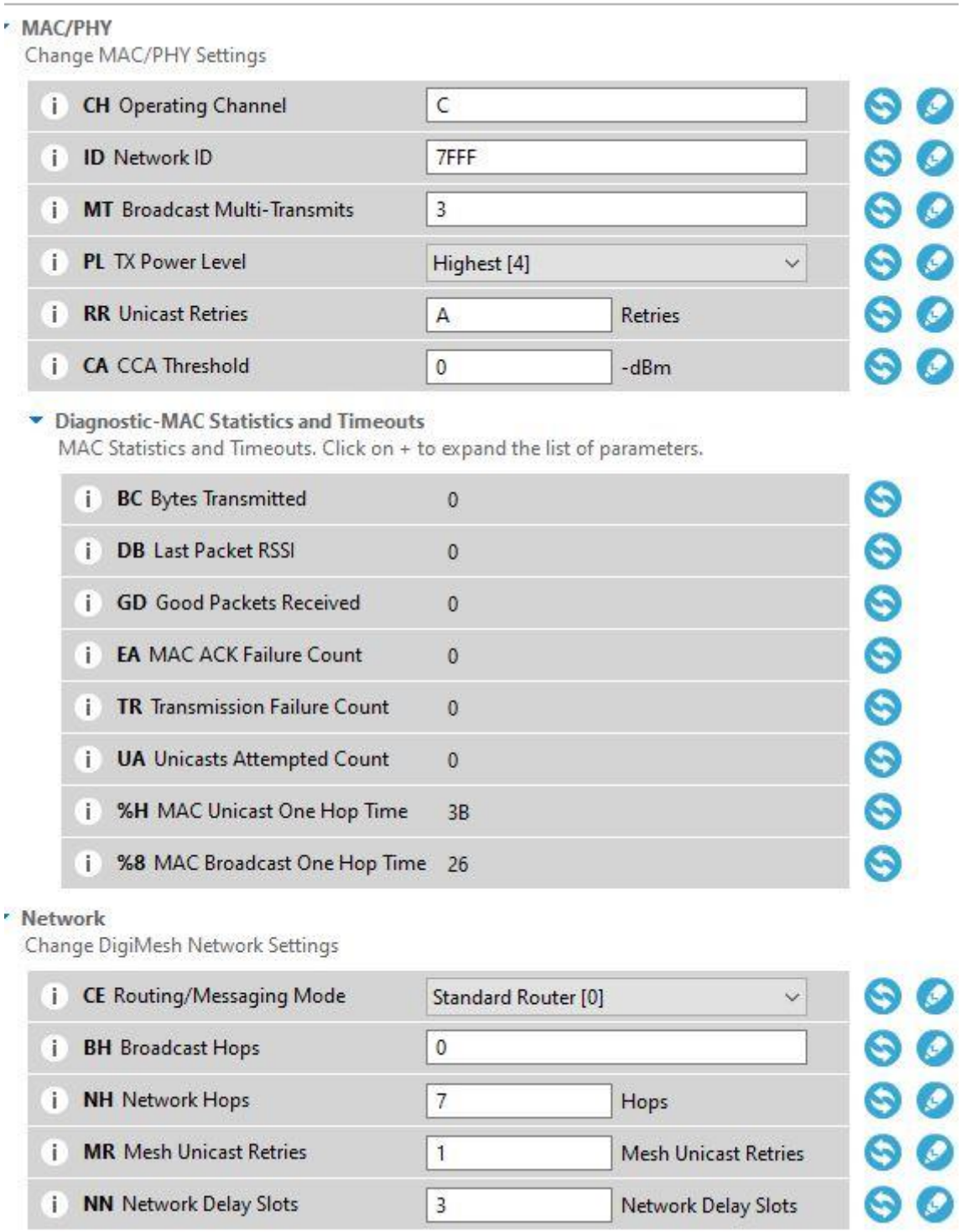


Figure 57.- Configuración Xbee 2

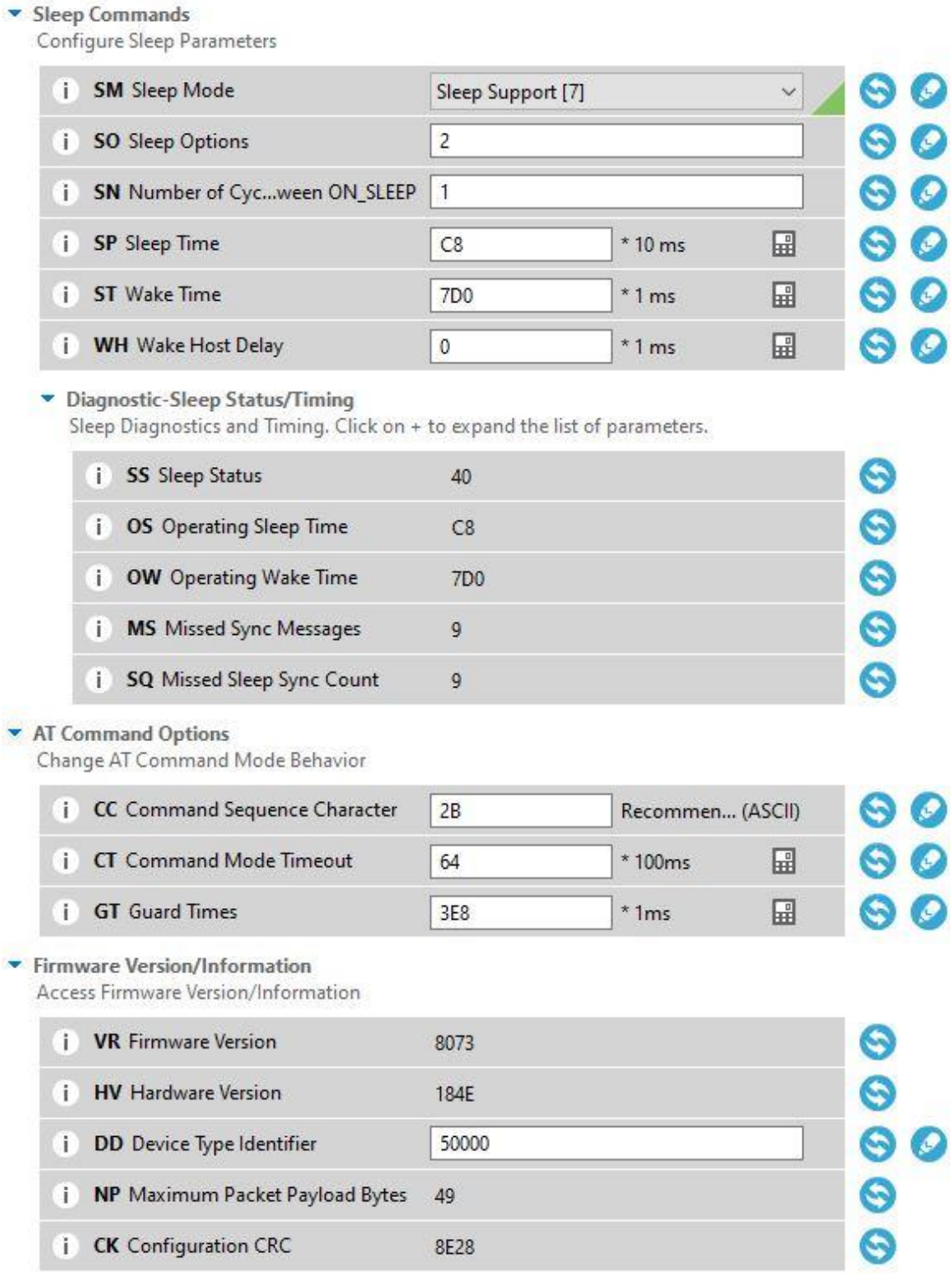


Figure 58.- Configuración Xbee 3

Existen dos diferencias principales entre esta configuración y las del resto, en esta, al ser la del coordinador, la dirección incluye en la parte baja todos los destinatarios posibles. En el caso de los nodos incorporaran la dirección del nodo router. En el caso

del router se incorporará la dirección del coordinador.
La segunda diferencia principal, está en el modo de sueño ya que el coordinador se encuentra en modo coordinación del sueño, SM7, mientras que el resto de nodos se encontraran en el modo SM8.

Es determinantemente importante que todos los nodos estén configurados con el mismo código de red.

También es importante que el coordinador y el router se encuentren en modo API, mientras que el resto de nodos pueden encontrarse tanto en modo api como en moto AT o transparente. Esto se produce por que las tramas en modo api integran además del mensaje la dirección del nodo que envia y algunos otros parámetros muy útiles para tratar los datos recibidos.

XCTU también incorpora el visor de comunicaciones i nos permite ver desde su consola los mensajes que recibe el módulo coordinador, en la siguiente imagen podemos ver un ejemplo.

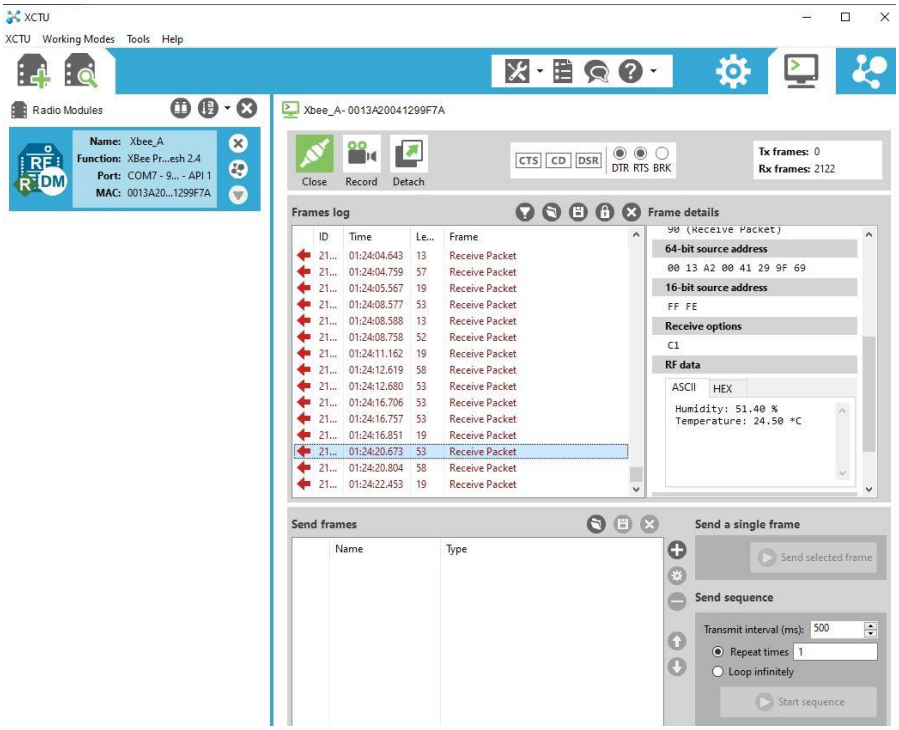


Figure 59.- Consola XCTU

4.4.2 SOFTWARE ARDUINO COORDINADOR

En el caso del arduino que se encuentra conectado al nodo xbee coordinador, incorpora un programa que adecua los datos recibidos y los reenvía a Matlab.

Es necesaria la implementación de esta funcionalidad en un dispositivo microprocesador con mínimo 2 puertos series hardware como, por ejemplo, el arduino MEGA. Esto es necesario ya que uno de los puertos lo ocupará el xbee para comunicar con el arduino y utilizaremos otro de los puertos para comunicar el arduino con el ordenador.

Aunque existe la posibilidad de crear en arduino puertos serie por software después de varias pruebas no dan el resultado esperado y se opta por una placa con varios puertos serie vía hardware.

El software que incluye el nodo coordinador incorpora 3 funcionalidades principales.

Principalmente recibe los datos de los nodos leyendo el puerto serie 1 y determina la longitud del mensaje para saber la longitud de la lectura, posteriormente lee todo el mensaje y realiza el cálculo del checksum. Una vez realizado este cálculo verifica que se corresponda con los últimos dos bytes del mensaje que incorporan este mismo dato dado por el emisor. Si el mensaje es correcto este mensaje mediante 8 tramas creadas por mí se envía a Matlab especificando:

- Trama 1: checksum
- Trama 2: checksum calculado
- Trama 3: Salto de línea
- Trama4: Longitud de la trama
- Trama5: Trama en Hexadecimal
- Trama6: Trama en ASCI (no es del todo correcta, pero permite ver parte del mensaje)
- Trama 7: Dirección del que envía el mensaje (64 bits)
- Trama 8: Tipo de menaje

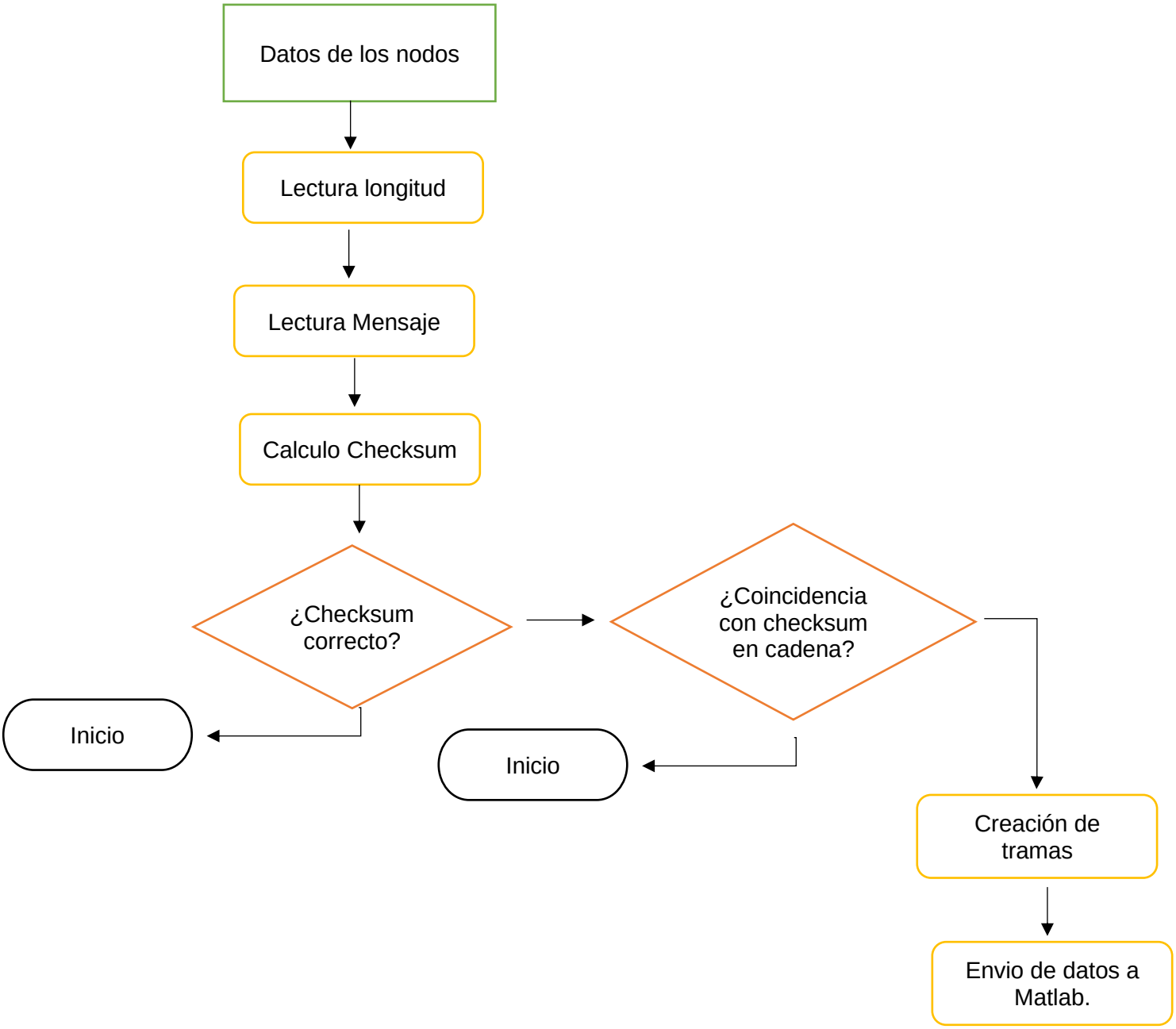
La segunda funcionalidad principal es la de enviar tramas de datos a los nodos periféricos, en este caso realiza la tarea complementaria a la anterior, recibe por el puerto serie 0 un mensaje de Matlab y lo envía al nodo destino.

Por último, la tercera funcionalidad es la recepción, después de un envío hacia un nodo, de la confirmación de recepción de un mensaje. Esta confirmación es la única manera de comprobar la correcta transmisión entre el coordinador y un nodo final.

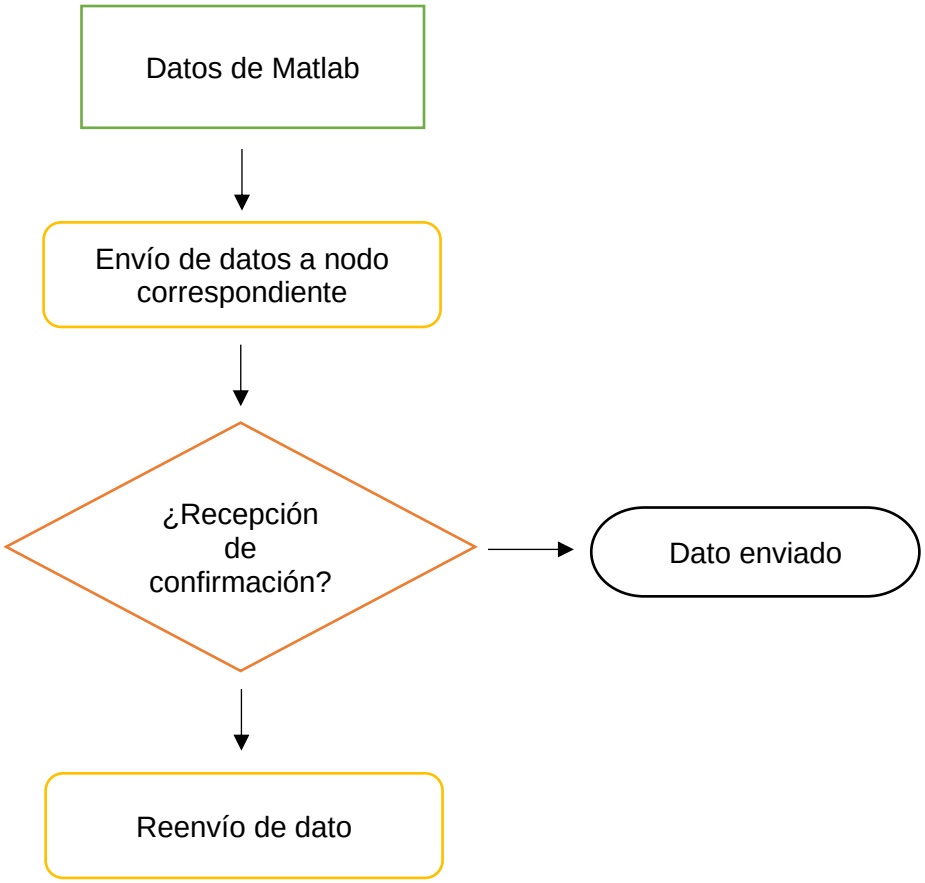
En el anexo 9.5, podemos ver el software del módulo arduino con algunos comentarios.

A continuación, podemos ver los flujogramas de este programa.

Este primer flujograma se corresponde con la primera funcionalidad del software del arduino coordinador.



A continuación, se incluye el flujograma de la segunda y la tercera funcionalidad.



Por el puerto serie arduino envía las siguientes tramas a Matlab:

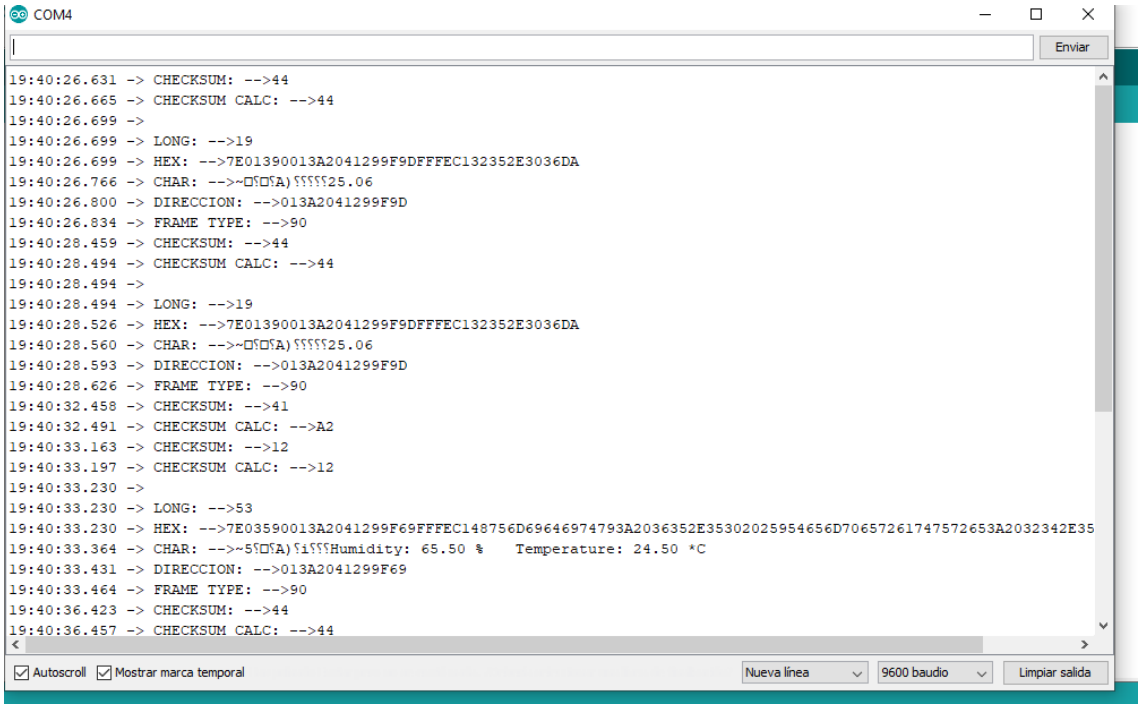


Figure 60.- Tramas de arduino coordinador a matlab

4.4.3 SOFTWARE ARDUINO EN NODOS FINALES

En el caso de la aplicación tenemos 3 nodos finales.
El primer nodo final es el nodo de sala y su programa de Matlab lee los valores del sensor por un pin y los envía mediante el puerto serie.

En el anexo 9.6, podemos ver el programa de adquisición de datos del sensor correspondiente al nodo de sala.

En el caso del segundo nodo final, el nodo máquina, su programa es muy similar al anterior en este caso solo enviamos el valor de la temperatura ya que siempre toma un dato. A diferencia del sensor del nodo sala que puede enviar error en la toma de datos.

En el anexo 9.7, podemos ver el programa de adquisición de datos del nodo máquina.

En la implementación de este sensor es muy importante analizar las velocidades de toma de datos de los sensores. En caso de trabajar en continuo es posible que si uno de los sensores tiene una velocidad el doble de rápida que otro sensor ocupe el canal serie y no permita la recepción de paquetes de otros nodos más lentos. Esto es posible solucionarlo añadiendo una pequeña pausa en aquellos sensores más rápidos o bien trabajando en tiempo discreto con los diferentes tiempos de adquisición controlados.

El ultimo nodo es el nodo periférico, en este nodo su ejemplificación es el control de un led. Este led simula un relé de accionamiento de una máquina.
A continuación, en el anexo 9.8, podemos ver el programa del nodo periférico.

4.5. ADQUISICION DE DATOS Y CONTROL

La adquisición de datos y el control se realizan mediante la herramienta matemática Matlab.

Usamos las funciones de Matlab para las diferentes acciones que queremos realizar incorporadas en una interfaz gráfica de tipo app generada en Matlab.

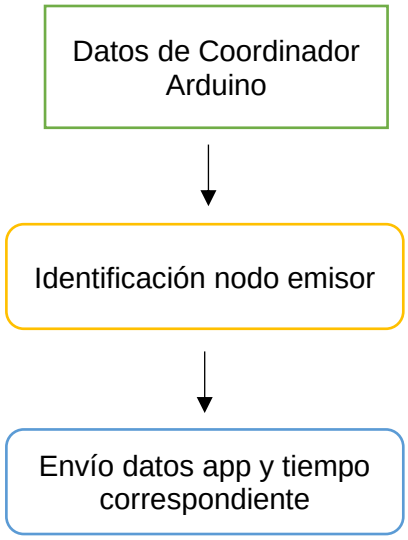
A diferencia de otras interfaces de visualización y alerta, esta app permite la interacción del usuario con el proceso además de la monitorización de los parámetros.

4.5.1 MATLAB

Para la realización de este prototipo se han utilizado 4 funciones de Matlab para 4 acciones diferentes.

En el primer caso la función Data_adq(), es la función que lee el puerto serie y envía los datos a la interfaz gráfica.

El código Matlab de esta función se puede ver comentado en el anexo 9.9.
A continuación, se incluye el flujograma de esta función.



Esta primera función es la más importante, lee los datos del puerto serie e identifica quien envía esos datos y los envía a la aplicación para graficarlos.

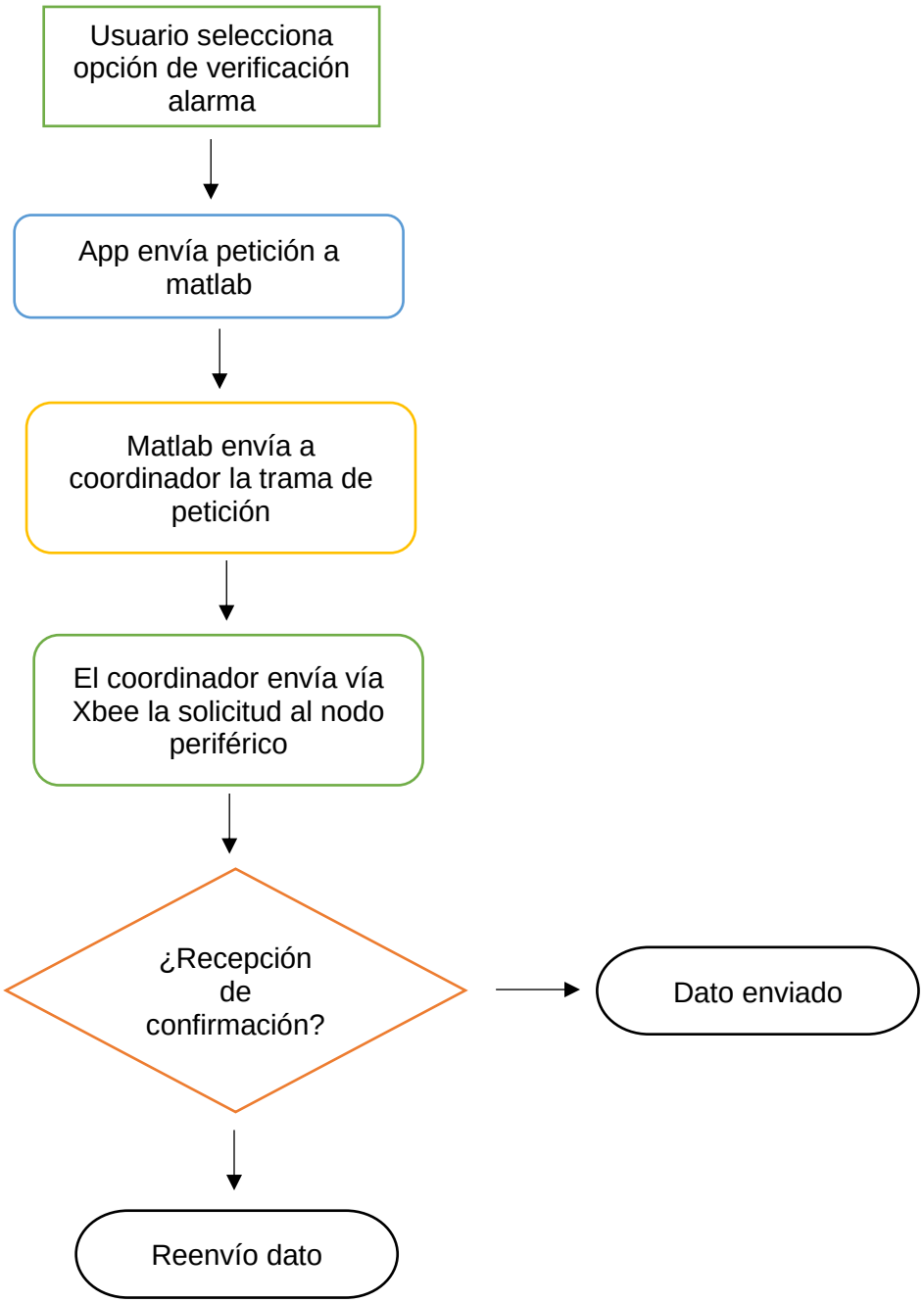
En este caso podemos ver las direcciones de los dos nodos que mandan valores y como se tratan sus cadenas. Se toman los valores y la función retorna el parámetro medido y el tiempo del gráfico en cuestión.

Esto es debido que, si usamos el mismo tiempo para todos los gráficos, en cada muestreo hay uno de los dos sensores que, para Matlab, toma una muestra en blanco ya que al ser una comunicación por un único canal no se puede recibir más de una trama simultáneamente.

Un factor determinante para lograr una buena comunicación tanto al enviar datos como al recibirlos es la necesidad de incorporar pausas después de la caracterización del puerto serie y después de abrir el puerto, así como antes de cerrarlo todo esto impedirá la perdida de datos o la recepción de caracteres diferentes a los deseados.

Además de esta función principal también están las funciones tiempo() y tiempo2(), cuya función es borrar los gráficos y resetear el tiempo de los gráficos en el momento que el usuario decida reiniciar el gráfico.

Por último la función Checkalarma() verifica el funcionamiento de la alarma enviando la trama correspondiente.
Podemos ver su código de Matlab en el anexo 9.10.



4.5.2 INTERFAZ GRAFICA

Para la usabilidad de esta aplicación, se ha decidido crear una aplicación para ordenador o teléfonos vía el software de Matlab App Designer que permite generar un entorno interactivo para las aplicaciones de Matlab y exportarlo como una aplicación.

La interfaz tiene el siguiente aspecto gráfico.

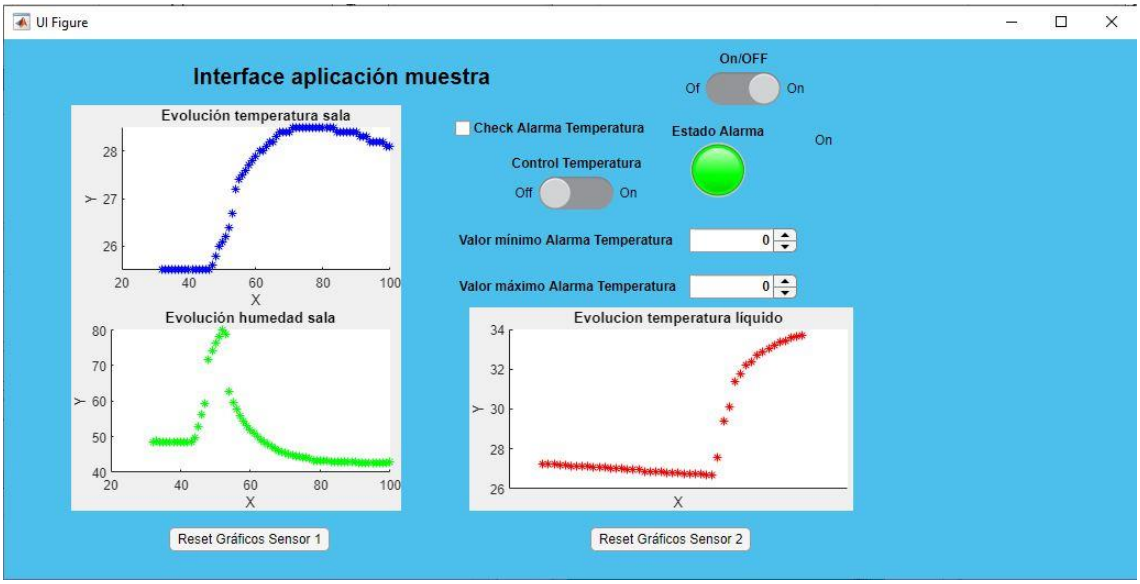


Figure 61.- Interfaz app prototipo

La interfaz gráfica está compuesta de 3 gráficos de monitorización de las muestras que llegan, en este caso dos referidos al nodo sala y uno referido al nodo máquina.

Además, incorpora la funcionalidad de control de la temperatura del nodo máquina pudiendo activar una alarma al superar un valor límite. Una vez sobrepasado este umbral se activa el led de alarma en la interfaz y se procede a la reactivación del periférico enviando la trama correspondiente al nodo de periférico.

También incorpora una opción de comprobación del funcionamiento de la alarma que realiza un test de 5 segundos del led alarma y el periférico.

El software detrás de esta interfaz se programa en lenguaje Matlab, y las acciones se incluyen en las funciones de los botones o bien en el propio inicio de la aplicación.

Por otro lado, aunque parece que tendría que ser muy sencillo, los espacios de variables no son conjuntos con Matlab y por tanto tenemos dos opciones para compartir variables de nuestro programa Matlab a la aplicación. La primera opción es la definición de algunas variables como globales, esta solución parecía la más sencilla, pero al probar bien su funcionalidad no acababa de funcionar bien siempre y dependiendo de si Matlab o la aplicación sobrescribían el valor de la variable se generaban errores internos.

Por tanto, surgió la necesidad de encontrar una segunda opción de incorporación de variables externas a la aplicación. Esta segunda opción era incorporar las variables como resultados de funciones de Matlab, ejecutándolas des de la aplicación retornaban los datos calculados en Matlab.

También incorpora la posibilidad de reiniciar cualquiera de los gráficos para que empiece a tomar datos desde 0 en aquel gráfico.

A continuación, en el anexo 9.11, se incorpora el código de la aplicación.

Podemos encontrar sombreada en gris la zona donde se realizan las tareas de llamar las funciones y de control mediante la interfaz.

La primera zona marcada en gris incorpora la lectura de los datos que entran en Matlab mediante la función de adquisición de datos.

En segundo lugar, encontramos las funciones que reinician los gráficos y su tiempo.

Y, por último, encontramos el trozo de programa encargado de hacer la comprobación de correcto funcionamiento de la alarma y la función que envía la alarma.

5. FUTURAS MEJORAS EN FASE DE DESARROLLO

En la actualidad se han configurado dos módulos en el entorno industrial y monitorizan datos de manera continua.
Debido a que todavía no se ha definido todos los datos necesarios para el correcto funcionamiento de la aplicación, por el departamento de ingeniería, el cálculo del consumo de los nodos queda pendiente.

Por otro lado, al ser un prototipo no se han realizado estudios de costes hasta que no se compruebe la correcta funcionalidad de los sensores implementados en un periodo de prueba. Es importante ver que no existen interferencias que no permitan la utilización de xbee como método de transmisión y obliguen a utilizar algún otro método de obtención de datos. Si pasado el periodo de prueba todo está correcto, se procederá a montar la instalación de sensores para una primera máquina, para a continuación implementarlo en toda la planta.

Al tratarse de un proyecto en empresa la parte física de los nodos no está definida todavía, ya que, a petición de la empresa, ésta será diseñada por el departamento de diseño y se adaptará a las necesidades precisas en cada lugar.

Por último, una vez implementado en la planta de inyección se estudiará la posibilidad de suministrar este producto a otras empresas.

6. CONCLUSIONES

Este trabajo contiene tres partes muy bien diferenciadas, cada una de ellas con unos objetivos asociados. En primer lugar, se considera que se ha generado una base de conocimiento suficiente para llevar a cabo proyectos de aplicaciones de IOT de bajo consumo analizando las opciones del mercado y eligiendo Xbee como la solución más adecuada debido a la gran cantidad de opciones que provee, al estar menos limitada que algunas de las tecnologías con las que compite.
En segundo lugar, la posibilidad de generar diferentes tipos de configuraciones red lo hacen de nuevo un producto más versátil y atractivo para una mayor cantidad de proyectos.

En lo que se refiere a la segunda parte del trabajo, se ha estudiado con éxito el hardware para las diferentes partes del proceso, desde la adquisición de los datos, pasando por su comunicación y procesado. Y se ha decidido tomar Arduino como base de los nodos y un ordenador para procesar todos los datos y controlar la aplicación. La elección de arduino es básicamente por la facilidad que supone su programación, su bajo coste y el reducido consumo de algunos de sus módulos, factor muy importante en aplicaciones de bajo consumo.

Referido a la última parte del proyecto, la elaboración de un prototipo que ejemplifique las casuísticas de la situación real en una fábrica, se ha conseguido testear con éxito todos los procedimientos necesarios para llevar a cabo la aplicación presentada.

Por todos estos factores satisfactorios, se puede concluir que este proyecto supone una buena base de un proyecto mucho más ambicioso, que se llevará a cabo en un entorno de aplicación industrial, destacándolo como uno de los factores principales de éxito para esta futura aplicación.

7. AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mis padres el apoyo en este proyecto.

En segundo lugar, me gustaría agradecer a mi empresa la oportunidad de desarrollar una base para llevar a cabo un proyecto real, y que desde un inicio estén siguiendo de manera expectante los avances del proyecto.

Por último, me gustaría agradecer a mi tutor de la universidad, Antonio Miguel López Martínez, haber confiado desde un principio en este proyecto.

8. BIBLIOGRAFIA

[1] DIGI. Disponible en:
https://www.digi.com/resources/documentation/Digidocs/90002002/Content/Reference/r_zb_stack.htm?TocPath=zigbee%20networks%7C_____3

[2] Zigbee. Disponible en:
www.ecured.cu/Topolog%C3%ADas_de_red_ZigBee

[3] Xbee. Disponible en:
www.digi.com/resources/documentation/digidocs/pdfs/90000982.pdf

[4] ZigBee Alliance. Disponible en:
www.zigbee.org

[5] Comparación low power. Disponible en:
accent-systems.com/es/blog/diferencias-nb-iot-lte-m

[6] SigFox. Disponible en:
www.sigfox.com/en

[7] Comunicaciones low power. Disponible en:
www.m2comm.co/front-page/technology/wan-ultra-narrow-band-unb

[8] IOT. Disponible en:
www.telensa.com/2015/08/22/why-ultra-narrow-band-unb-is-rewriting-the-economics-of-iot

[9] Aplicaciones IOT. Disponible en:
www.forbes.com.mx/mexico-estrena-red-para-el-internet-de-las-cosas

[10] LoRa One Disponible en:
www.link-labs.com/blog/what-is-lora

[11] Comparación protocolos inalámbricos. Disponible en:
www.lifewire.com/guide-to-wireless-network-protocols-817966

[12] Comparación Comunicaciones bajo consumo Disponible en:
signalsiot.com/lora-cat-m-y-nb-iot-como-escoger-el-protocolo-iot-adecuado

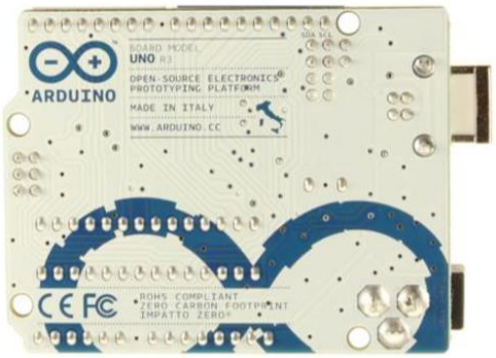
[13] Xbee Disponible en:
xbee.cl

9. ANNEXOS

9.1. DATA SHEET ARDUINO UNO

Arduino Uno

Arduino Uno R3 Front

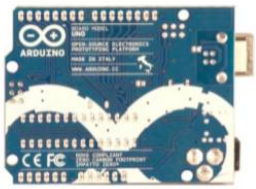


Arduino Uno R3 Back

Arduino Uno R2 Front

Arduino Uno SMD

Arduino Uno Front



Arduino Uno Back

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.
Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.
Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V

Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: arduino-uno-Rev3-reference-design.zip (NOTE: works with Eagle 6.0 and newer)
Schematic: arduino-uno-Rev3-schematic.pdf
Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.
The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.**This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.
The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.
The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.
The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

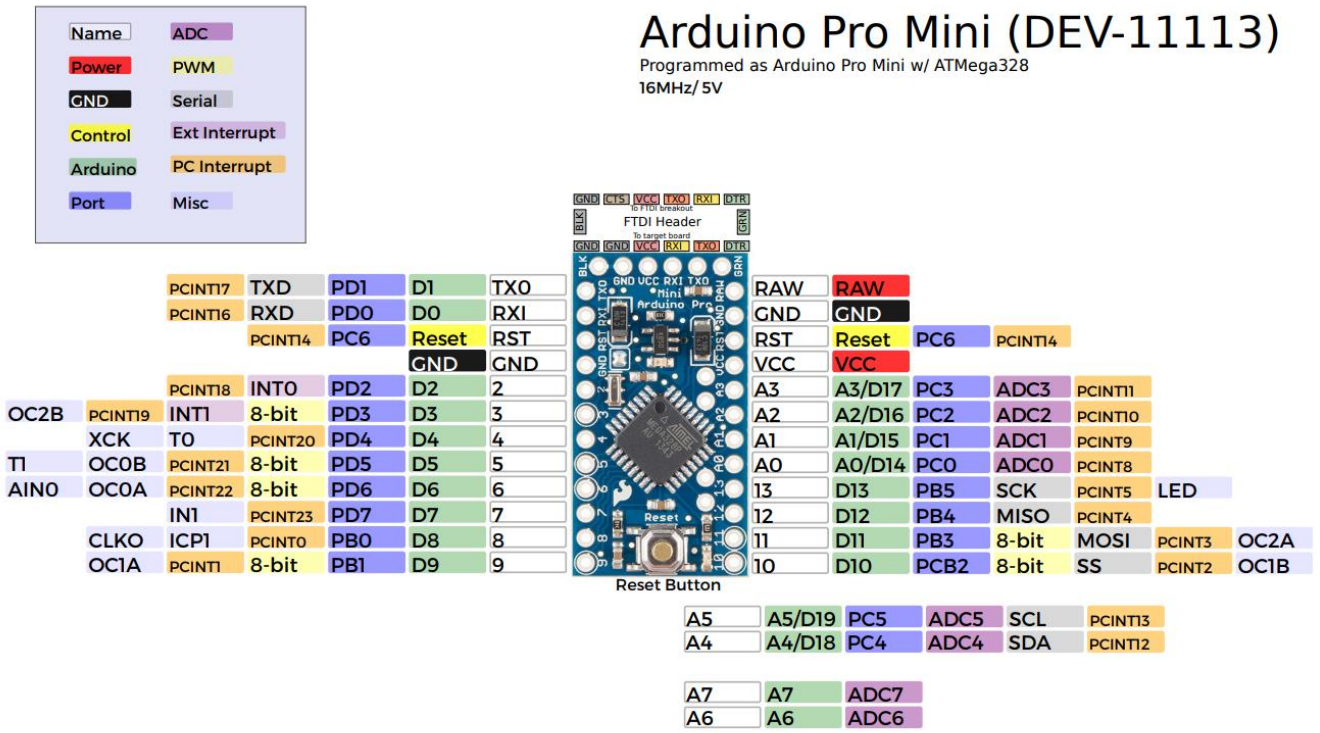
USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

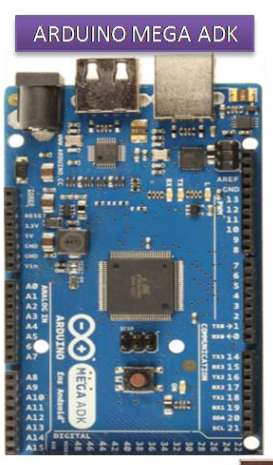
Physical Characteristics

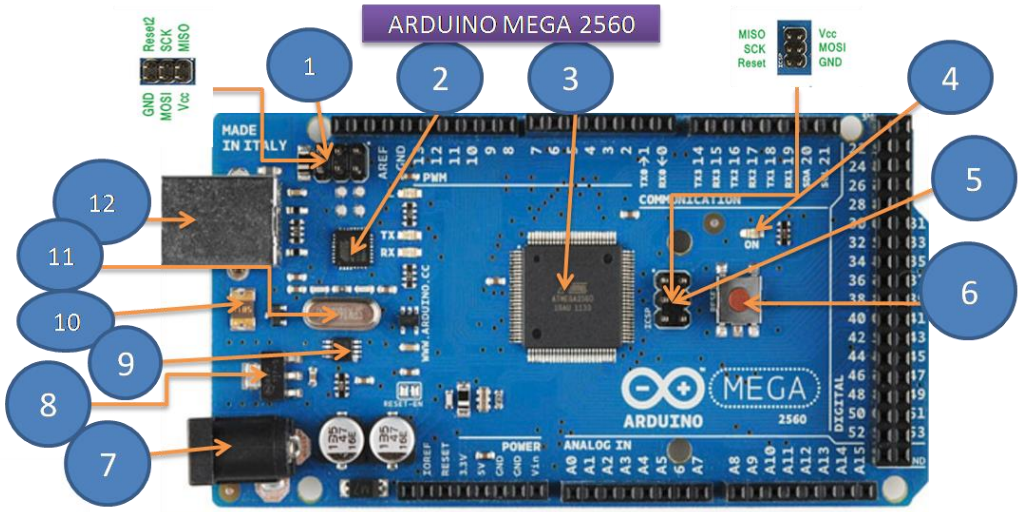
The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

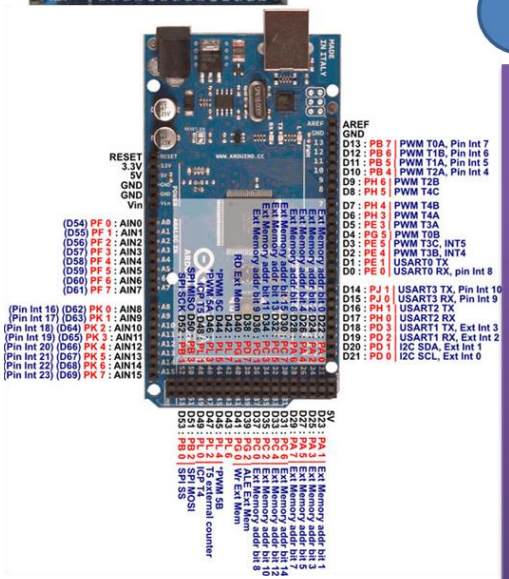
9.2. DATA SHEET ARDUINO MINI PRO



9.3. DATA SHEET ARDUINO MEGA







- Microcontroller- ATmega2560
- Operating Voltage- 5V
- Input Voltage (recommended)- 7-12V
- Input Voltage (limits)- 6-20V
- Digital I/O Pins- 54 (of which 15 provide PWM output)
- Analog Input Pins- 16
- DC Current per I/O Pin- 40 mA
- DC Current for 3.3V Pin- 50 mA
- Flash Memory- 256 KB of which 8 KB used by bootloader
- SRAM- 8 KB
- EEPROM- 4 KB
- Clock Speed- 16 MHz
- USB Host Chip- MAX3421E
- Length- 101.52 mm
- Width- 53.3 mm
- Weight- 36 g

1-ICSP for 16u2 USB interface
2-Serial communication chip
3-Atmega 2560
4-PowerLED
5-ICSP for Atmega 2560
6-Reset button
7-7v to 12v DC input ,center positive
8-Voltage Regulator 5V
9-Voltage Regulater3.3V
10-Over current protector
11-Clockspeed/Crystal 16MHz
12-USB plug to computer

9.4. DATA SHEET BATERIA ER34615M

ULTRALIFE
BATTERY & ENERGY PRODUCTS

ER34615M: D Size Spiral Cell
Technical Datasheet

Features

- High and flat operating voltage
- High power and higher energy for the whole battery life
- High drain capability
- Higher power applications
- Low self-discharge rate (<1% per year at 20°C)
- Battery life higher than 10 years, depending on the application
- Hermetic glass-to-metal sealing
- Avoid leakage, key for higher than 10 year battery life
- Non-flammable electrolyte
- Safer operation in case of abuse
- PTC device
- Safe operation in the event of a short circuit

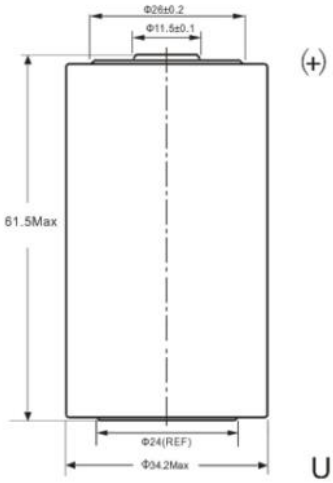
Typical Applications

- Military and other radio applications

Technical Specifications	
Part No.	ER34615M (UHR-ER34615)
Cell Type	Primary, non-rechargeable
Chemistry	Lithium Thionyl Chloride
Voltage Range	2.0 to 3.7V
Nominal Voltage	3.6V
Nominal Capacity	14.5Ah @ 10mA to 2.0V @ 20°C
Max. Continuous Discharge Current	2000mA continuous
Max. Pulse Discharge Current	Up to 3000mA (life and temperature dependent)
Weight	107g
Operating Temperature	-55°C to 70°C 70°C to 85°C (max of 24 hours)
Storage Temperature [1]	-55°C to 70°C 70°C to 85°C (max of 24 hours)
Exterior/Housing	Stainless steel container
Terminals/Connector	Radial tabs / radial pins / axial leads / flying leads
Safety	AL-MSDS/RD-002 Material Safety Datasheet - MSDS041 Safety Guide UBM-5112
Transportation [2]	Class 9 - A complete description of transportation regulations, lithium weights and transportation classifications is available on the Ultralife website.
Quality Assurance	Ultralife manufacturing facilities are ISO 9001:2008 and ISO 14001:2004 registered. Its products are listed under the Component Recognition Program of Underwriters Laboratories (UL) and have passed UN transportation testing, which is required for international transportation of all lithium batteries.

Notes
1. Cells should be stored in temperatures less than 30°C for a shelf-life of greater than five years.

Dimensions



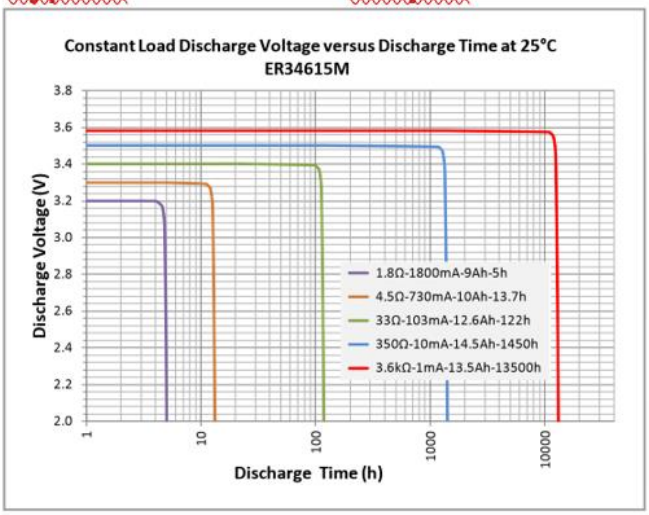
Unit:mm

2. A complete description of transportation regulations, lithium weights and transportation classifications is available on the Ultralife website.

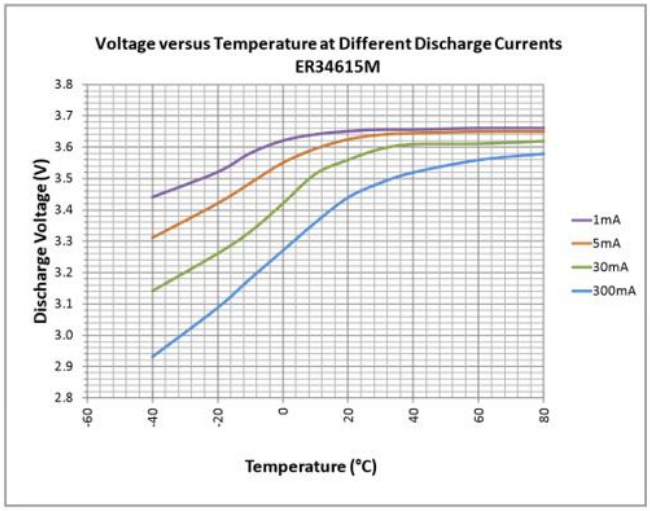
- Smart meters
- Alarm and security systems
- Beacons and emergency location transmitters
- GPS
- LED lighting applications

©2018 Ultralife Corporation • www.ultralifecorp.com • All information is subject to change without notice.
The information contained herein is for reference only and does not constitute a warranty of performance. • 9 MAY 18 UBM-0068 Rev: C

Typical Performance Graphs



High and flat voltage at high and low drain



High voltage at high drain even at -30°C

9.5. CÓDIGO ARDUINO COORDINADOR

```
long incomingByte = 0; // entrada de datos de nodos
long incomingByteNext = 0;//
long incomingByte2=0;// entrada de datos desde matlab
long incomingByte3=0;// entrada de recepción de confirmación de mensajes
byte msg [100];
byte msg2 [50];
byte valor;
byte valor2;
char valor3;
int Index=0;
byte CodigoInicio=0x7E;
bool started= false;//True: mensaje iniciado
bool ended   = false;//True: mensaje acabado
int Longitud;
String LongitudSt;
byte Direccion [8];
long LongitudH;
long LongitudL;
byte FrameType;
byte CheckSum;
int CheckSumCalc;


void setup() {
  Serial1.begin(9600); // opens serial port, sets data rate to 9600 bps
  Serial.begin(9600);
  Serial2.begin(9600);

}

void loop() {
  //mirem el que arriba de matlab i ho enviem per xbee
    if(Serial.available() > 0){
      delay(50);
     incomingByte2 = Serial.read();
     Serial2.print(incomingByte2);
     }
    //mirem el que ens arriba de rele , confirmacio
     if(Serial2.available() > 0){
      delay(50);
      for(int i=0;i<4;i++){
         while(Serial2.available() == 0){
       }
       incomingByte3 = Serial2.read();
       msg2[i]= incomingByte3;}
        Serial.println(incomingByte3);
        for(int i=0;i<4;i++){ Serial.print(msg2[i],HEX);}
     }
```

```
//Lectura de dades de nodes y envio a matlab:
while (Serial1.available() > 0) {
  // read the incoming byte:
    // if (started == false)
    //                  {incomingByte = Serial1.read();
    //                   //incomingByteNext = Serial1.read();
    //                   started == true;
    //                  }
    // else               {incomingByte = incomingByteNext;
    //                  // incomingByteNext = Serial1.read();
    //                  }
     incomingByte = Serial1.read();

  if (incomingByte == CodigoInicio && Index==0)
  {
    started = true;
    Index = 0;
    // Borra msg
    for(int i=0;i<99;i++){msg[i]=0;   }
     //Serial.println(" ");
     //Serial.println("INICIAL");
     //Serial.print(Index);
     //Serial.print(":");
     //--Serial.println(incomingByte,HEX);//imprimir el caracter 7E
     msg[Index]=incomingByte; // Add char to array
     Index++;
     // Leer Longitud
     while(Serial1.available() == 0){
     }
              LongitudH = Serial1.read();
             //-- Serial.print(Index);
             //-- Serial.print(":");
             //-- Serial.println(LongitudH,HEX);//imprimir parte alta longitud
              msg[Index]=LongitudH; // Add char to array
              Index++;
    while(Serial1.available() == 0){
    }
              LongitudL = Serial1.read();
             //-- Serial.print(Index);
             //-- Serial.print(":");
             //-- Serial.println(LongitudL,HEX);// imprimir parte baja longitud
              msg[Index]=LongitudL; // Add char to array
              Index++;
    //calculem la longitud
    LongitudSt = msg[1] + msg[2];//encadenem els dos valors un rere laltre
    Longitud = LongitudSt.toInt();//pasem el numero sencer a enter
  //-- Serial.println(" ");
  //-- Serial.print("LONG: -->");
  //-- Serial.println(Longitud);//mostrem per pantalla el valor de la longitud
```

```
  }
    else if(Index == Longitud+4)//imprimim els resultats quan hem llegit tot el misatje
{
      //CheckSum
   CheckSum = msg [Longitud+ 3];
   Serial.print("CHECKSUM: -->");
   Serial.println(CheckSum,HEX);

   //Calcular CheckSum
   CheckSumCalc  = 0;
   for (int i = 3; i< (Longitud + 3); i=i+1) {CheckSumCalc = CheckSumCalc + msg[i];}
   CheckSumCalc = CheckSumCalc & 0x00FF;
   CheckSumCalc = 0xFF - CheckSumCalc;
    Serial.print("CHECKSUM CALC: -->");
   Serial.println(CheckSumCalc ,HEX);
   //mirem si el check sum  es correcte, nomes imprimim si check sum correcte
   if (CheckSumCalc == CheckSum ){
   Serial.println(" ");
  Serial.print("LONG: -->");
  Serial.println(Longitud);//mostrem per pantalla el valor de la longitud
  //-- Serial.println(" ");
   //--Serial.println("FI");
   Serial.print("HEX: -->");
// Message in Hexadecimal
   for(int i=0;i<(Longitud + 3);i++){ Serial.print(msg[i],HEX);}
   Serial.println(" ");
   Serial.print("CHAR: -->");
// Message in Char
   for(int i=0;i<(Longitud + 3);i++){valor3 = msg[i];
                        Serial.print(valor3);
                       }
 // Direcciones
    //Serial.println("-- ");
    for (int i = 0; i< 8; i=i+1) { Direccion[i] = msg[i+4];}
    Serial.print("DIRECCION: -->");

    for(int i=0;i<8;i++){ Serial.print(Direccion[i],HEX);}
   Serial.println(" ");
 //FrameType
   FrameType = msg[3];
   Serial.print("FRAME TYPE: -->");
   Serial.println(FrameType,HEX);
   }

  started = false;
  Index = 0;
  break; // lectura hecha
}
//bucle lectura del mensaje
else
{
  if(Index < 99 && started == true ) // asegurarse de espacio en vector
  {  //--Serial.print(Index);
```

```
        //--Serial.print(":");
        //--Serial.print(incomingByte,HEX);
        //--Serial.print(" ");
        valor3 = incomingByte;
       //-- Serial.println(valor3);
       msg[Index]=incomingByte; // incluir char en array
       Index++;
     }
   }
 }
 }
```

9.6. CÓDIGO ARDUINO NODO SALA

```
#include "DHT.h"

#define DHTPIN 2     // Pin donde esta conectado el sensor

#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

void setup()
{

   SERIAL.begin(9600);
   SERIAL.println("DHTxx test!");
   Wire.begin();

   /*if using WIO link,must pull up the power pin.*/
   // pinMode(PIN_GROVE_POWER, OUTPUT);
   // digitalWrite(PIN_GROVE_POWER, 1);

   dht.begin();
}

void loop()
{
   float temp_hum_val[2] = {0};

   if(!dht.readTempAndHumidity(temp_hum_val)){
       SERIAL.print("Humidity: ");
       SERIAL.print(temp_hum_val[0]);
       SERIAL.print(" %\t");
       SERIAL.print("Temperature: ");
       SERIAL.print(temp_hum_val[1]);
       SERIAL.println(" *C");
   }
   else{
      SERIAL.println("Failed to get temprature and humidity value.");
   }
```

```
   delay(1500);
}
```

9.7. CÓDIGO ARDUINO NODO DE MÁQUINA

```
#include <OneWire.h>
#include <DallasTemperature.h>

// sensor conectat al pin 2 de Arduino
#define ONE_WIRE_BUS 2

OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);
String Temperatura;

void setup(void)
{
  // start serial port
  Serial.begin(9600);
  //Serial.println("Dallas Temperature IC Control Library Demo");

  // Start up the library
  sensors.begin();
}


void loop(void)
{
  sensors.requestTemperatures(); // comando de lectura de temperatura
  //Serial.print("Temperature is:");
  Serial.println(sensors.getTempCByIndex(0)); // Why "byIndex"?
    // You can have more than one IC on the same bus.

    delay(5000);
}
```

9.8. CÓDIGO ARDUINO NODO PERIFÉRICO

```
long lectura=0;
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode(3,OUTPUT);


}
```

```
void loop() {
  // put your main code here, to run repeatedly:
 if(Serial.available() > 0){
      delay(50);
       lectura = Serial.read();

        Serial.println(lectura);
       digitalWrite(13,HIGH);
       delay(5000);
         digitalWrite(13,LOW);
 }
}
```

9.9. CÓDIGO MATLAB FUNCIÓN DATA_ADQ()

```
function [t2,t,T2,TS2,TH2]= Data_adq()
    chr = blanks (150);
    n=true;
    hold on %para que la gráfica no se borre
    global t; %tiempo 0
    global t2;
      if isempty(t2)
         t2=0;
    end

    if isempty(t)
        t=0;
    end

    T2=0;
    TS2=0;
    TH2=0;
    chr7=blanks(100);
    %Configuracion puerto Serie
    delete(instrfind({'port'},{'COM3'}));
    PuertoSerie = serial('COM3'); %Puerto que leer
    PuertoSerie.BaudRate = 9600; %Velocidad de transmisión
    %f=uifigure(app1)
    fopen(PuertoSerie); %Abrir el puerto
    chr = fscanf(PuertoSerie)%leemos las diferentes cadenas de mi trama
    chr2= fscanf(PuertoSerie)
    chr3= fscanf(PuertoSerie)
    chr4=fscanf(PuertoSerie)
    chr5=fscanf(PuertoSerie)
    chr6=fscanf(PuertoSerie)
    chr7=fscanf(PuertoSerie)
    chr8=fscanf(PuertoSerie)
    fclose(PuertoSerie); %cerrar el puerto
      if strfind(chr7,'013A2041299F9D') %direccion nodo maquina
        t=t+1; %aumentar en uno el tiempo
        T=strcat(chr6(25),chr6(26),chr6(27),chr6(28),chr6(29)) %Temperatura
        T2=str2num(T)
```

```
      %plot(t,T2,'-*') %dibujar el punto de la temperatura en el tiempo que va
      pause(1) %pausar el tiempo un segundo
      %t=t+1; %aumentar en uno el tiempo
    end
    if strfind(chr7,'013A2041299F69')% dirección nodo sala
       if strfind(chr6,'Temperature:')% eliminamos las cadenas  fail temperature
         t2=t2+1;
         TS=strcat(chr6(56),chr6(57),chr6(58),chr6(59),chr6(60))
         TS2=str2num(TS)
         TH=strcat(chr6(35),chr6(36),chr6(37),chr6(38),chr6(39))
         TH2=str2num(TH)
         % plot(t,TS2,'-*')
         % plot(t,TH2,'-*')%dibujar el punto de la temperatura en el tiempo que va
         pause(1) %pausar el tiempo un segundo
         %t=t+1; %aumentar en uno el tiempo
       end
    end
end
```

9.10. CÓDIGO MATLAB FUNCIÓN CHECKALARMA()

```
function CheckAlarma()
%Configuracion puerto Serie
  x=serial('COM3','BAUD',9600);
  pause(1);
   fopen(x); %Abrir el puerto
    pause(1);
   Alarma="Alarma";
    fprintf(x,'%x',"Alarma")
     pause(2);
   fclose(x);%cerrar el puerto
end
```

9.11. CÓDIGO MATLAB APP ENTORNO GRÁFICO

```
classdef app3 < matlab.apps.AppBase

   % Properties that correspond to app components
   properties (Access = public)
      UIFigure                    matlab.ui.Figure
      UIAxes                      matlab.ui.control.UIAxes
      UIAxes_2                    matlab.ui.control.UIAxes
      ResetGrficosSensor1Button   matlab.ui.control.Button
      ResetGrficosSensor2Button   matlab.ui.control.Button
      UIAxes_3                    matlab.ui.control.UIAxes
      InterfaceaplicacinmuestraLabel  matlab.ui.control.Label
      ValormnimoAlarmaTemperaturaSpinnerLabel  matlab.ui.control.Label
      ValormnimoAlarmaTemperaturaSpinner  matlab.ui.control.Spinner
      EstadoAlarmaLampLabel       matlab.ui.control.Label
      EstadoAlarmaLamp            matlab.ui.control.Lamp
```

```
    ControlTemperaturaSwitchLabel   matlab.ui.control.Label
    ControlTemperaturaSwitch        matlab.ui.control.Switch
    OnOFFSwitchLabel                matlab.ui.control.Label
    OnOFFSwitch                     matlab.ui.control.Switch
    Label                           matlab.ui.control.Label
    CheckAlarmaButton               matlab.ui.control.Button
    Label_2                         matlab.ui.control.Label
    T2Label                         matlab.ui.control.Label
    Label_3                         matlab.ui.control.Label
    CLabel                          matlab.ui.control.Label
end

properties (Access = public)
    Property2 % Description
end



% Callbacks that handle component events
methods (Access = private)

    % Code that executes after component creation
    function startupFcn(app)

    end

    % Callback function
    function ResetGrficosSensor1ButtonPushed(app, event)

    end

    % Callback function
    function ResetGrficosSensor1ButtonPushed2(app, event)

    end

    % Callback function
    function ActivargraficosButtonPushed(app, event)

        %app1.ejex1.value=0;
    end

    % Value changed function: OnOFFSwitch
    function OnOFFSwitchValueChanged(app, event)
        value = app.OnOFFSwitch.Value;
        app.Label.Text=num2str(value);
      while (value == 'On')
          %global t
          % global T2
           [t2,t,T2,TS2,TH2]=Data_adq();
           % app.Label_2.Text=num2str(t)
           hold(app.UIAxes_2,'on');
           if T2~=0
```

```
            plot(app.UIAxes_2,t,T2,'-*r');
            app.Property2=T2;
            app.Label_3.Text=num2str(T2);
            end
           value = app.OnOFFSwitch.Value;
           %grafic temperatura i humitat
            hold(app.UIAxes,'on');
            if TS2~=0
            plot(app.UIAxes,t2,TS2 ,'-*b');
            end
            %humitat
            hold(app.UIAxes_3,'on');
            if TH2~=0
            plot(app.UIAxes_3,t2,TH2,'-*g');
            end

           value = app.OnOFFSwitch.Value;
           value2 = app.ControlTemperaturaSwitch.Value;
           app.Label_2.Text=num2str(value2);
        if (value2 =='On')
           if app.Property2> app.ValormnimoAlarmaTemperaturaSpinner.Value
            CheckAlarma()
            app.EstadoAlarmaLamp.Color='red';
           end
           if app.Property2< app.ValormnimoAlarmaTemperaturaSpinner.Value
            app.EstadoAlarmaLamp.Color='green';
           end
        end
        if (value2 =='Of')
          app.EstadoAlarmaLamp.Color='green';
        end
        end
        %while value == 'No'
           % hold(app.UIAxes_2,'off');

            %value = app.OnOFFSwitch.Value;


    end

    % Button pushed function: ResetGrficosSensor1Button
    function ResetGrficosSensor1ButtonPushed3(app, event)
     timepo0();
      cla (app.UIAxes);
       cla (app.UIAxes_3);
    end

    % Button pushed function: ResetGrficosSensor2Button
    function ResetGrficosSensor2ButtonPushed(app, event)
       tiempo02();
     cla (app.UIAxes_2);
    end
```

```
        % Button pushed function: CheckAlarmaButton
        function CheckAlarmaButtonPushed(app, event)
            CheckAlarma();
            app.EstadoAlarmaLamp.Color='red';
            pause(5);
            app.EstadoAlarmaLamp.Color='green';
        end

        % Value changed function: ControlTemperaturaSwitch
        function ControlTemperaturaSwitchValueChanged(app, event)

        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Color = [0.302 0.749 0.9294];
            app.UIFigure.Position = [100 100 1030 491];
            app.UIFigure.Name = 'UI Figure';

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'Evolución temperatura sala')
            xlabel(app.UIAxes, 'X')
            ylabel(app.UIAxes, 'Y')
            app.UIAxes.Position = [62 247 300 185];

            % Create UIAxes_2
            app.UIAxes_2 = uiaxes(app.UIFigure);
            title(app.UIAxes_2, 'Evolucion temperatura líquido')
            xlabel(app.UIAxes_2, 'X')
            ylabel(app.UIAxes_2, 'Y')
            app.UIAxes_2.XTick = [0 0.2 0.4 0.6 0.8 1];
            app.UIAxes_2.Position = [424 63 349 185];

            % Create ResetGrficosSensor1Button
            app.ResetGrficosSensor1Button = uibutton(app.UIFigure, 'push');
            app.ResetGrficosSensor1Button.ButtonPushedFcn = createCallbackFcn(app,
@ResetGrficosSensor1ButtonPushed3, true);
            app.ResetGrficosSensor1Button.Position = [151 26 146 22];
            app.ResetGrficosSensor1Button.Text = 'Reset Gráficos Sensor 1';

            % Create ResetGrficosSensor2Button
            app.ResetGrficosSensor2Button = uibutton(app.UIFigure, 'push');
            app.ResetGrficosSensor2Button.ButtonPushedFcn = createCallbackFcn(app,
@ResetGrficosSensor2ButtonPushed, true);
            app.ResetGrficosSensor2Button.Position = [534 26 146 22];
```

```
        app.ResetGrficosSensor2Button.Text = 'Reset Gráficos Sensor 2';

        % Create UIAxes_3
        app.UIAxes_3 = uiaxes(app.UIFigure);
        title(app.UIAxes_3, 'Evolución humedad sala')
        xlabel(app.UIAxes_3, 'X')
        ylabel(app.UIAxes_3, 'Y')
        app.UIAxes_3.Position = [62 63 300 185];

        % Create InterfaceaplicacinmuestraLabel
        app.InterfaceaplicacinmuestraLabel = uilabel(app.UIFigure);
        app.InterfaceaplicacinmuestraLabel.FontSize = 20;
        app.InterfaceaplicacinmuestraLabel.FontWeight = 'bold';
        app.InterfaceaplicacinmuestraLabel.Position = [173 444 279 28];
        app.InterfaceaplicacinmuestraLabel.Text = 'Interface aplicación muestra';

        % Create ValormnimoAlarmaTemperaturaSpinnerLabel
        app.ValormnimoAlarmaTemperaturaSpinnerLabel = uilabel(app.UIFigure);
        app.ValormnimoAlarmaTemperaturaSpinnerLabel.HorizontalAlignment = 'right';
        app.ValormnimoAlarmaTemperaturaSpinnerLabel.FontWeight = 'bold';
        app.ValormnimoAlarmaTemperaturaSpinnerLabel.Position = [414 282 201 22];
        app.ValormnimoAlarmaTemperaturaSpinnerLabel.Text = 'Valor mínimo Alarma
Temperatura';

        % Create ValormnimoAlarmaTemperaturaSpinner
        app.ValormnimoAlarmaTemperaturaSpinner = uispinner(app.UIFigure);
        app.ValormnimoAlarmaTemperaturaSpinner.FontWeight = 'bold';
        app.ValormnimoAlarmaTemperaturaSpinner.Position = [630 282 100 22];

        % Create EstadoAlarmaLampLabel
        app.EstadoAlarmaLampLabel = uilabel(app.UIFigure);
        app.EstadoAlarmaLampLabel.HorizontalAlignment = 'right';
        app.EstadoAlarmaLampLabel.FontWeight = 'bold';
        app.EstadoAlarmaLampLabel.Position = [602 397 90 22];
        app.EstadoAlarmaLampLabel.Text = 'Estado Alarma';

        % Create EstadoAlarmaLamp
        app.EstadoAlarmaLamp = uilamp(app.UIFigure);
        app.EstadoAlarmaLamp.Position = [625 348 50 50];

        % Create ControlTemperaturaSwitchLabel
        app.ControlTemperaturaSwitchLabel = uilabel(app.UIFigure);
        app.ControlTemperaturaSwitchLabel.HorizontalAlignment = 'center';
        app.ControlTemperaturaSwitchLabel.FontWeight = 'bold';
        app.ControlTemperaturaSwitchLabel.Position = [460 368 123 22];
        app.ControlTemperaturaSwitchLabel.Text = 'Control Temperatura';

        % Create ControlTemperaturaSwitch
        app.ControlTemperaturaSwitch = uiswitch(app.UIFigure, 'slider');
        app.ControlTemperaturaSwitch.Items = {'Of', 'On'};
        app.ControlTemperaturaSwitch.ValueChangedFcn = createCallbackFcn(app,
@ControlTemperaturaSwitchValueChanged, true);
        app.ControlTemperaturaSwitch.Position = [487 337 69 30];
```

```
            app.ControlTemperaturaSwitch.Value = 'Of';

            % Create OnOFFSwitchLabel
            app.OnOFFSwitchLabel = uilabel(app.UIFigure);
            app.OnOFFSwitchLabel.HorizontalAlignment = 'center';
            app.OnOFFSwitchLabel.FontWeight = 'bold';
            app.OnOFFSwitchLabel.Position = [649 463 49 22];
            app.OnOFFSwitchLabel.Text = 'On/OFF';

            % Create OnOFFSwitch
            app.OnOFFSwitch = uiswitch(app.UIFigure, 'slider');
            app.OnOFFSwitch.Items = {'Of', 'On'};
            app.OnOFFSwitch.ValueChangedFcn = createCallbackFcn(app,
@OnOFFSwitchValueChanged, true);
            app.OnOFFSwitch.Position = [639 432 69 30];
            app.OnOFFSwitch.Value = 'Of';

            % Create Label
            app.Label = uilabel(app.UIFigure);
            app.Label.Position = [729 461 35 22];

            % Create CheckAlarmaButton
            app.CheckAlarmaButton = uibutton(app.UIFigure, 'push');
            app.CheckAlarmaButton.ButtonPushedFcn = createCallbackFcn(app,
@CheckAlarmaButtonPushed, true);
            app.CheckAlarmaButton.Position = [456 410 100 22];
            app.CheckAlarmaButton.Text = 'Check Alarma';

            % Create Label_2
            app.Label_2 = uilabel(app.UIFigure);
            app.Label_2.Position = [504 303 35 22];

            % Create T2Label
            app.T2Label = uilabel(app.UIFigure);
            app.T2Label.Position = [813 411 25 22];
            app.T2Label.Text = 'T2:';

            % Create Label_3
            app.Label_3 = uilabel(app.UIFigure);
            app.Label_3.Position = [837 410 35 22];

            % Create CLabel
            app.CLabel = uilabel(app.UIFigure);
            app.CLabel.Position = [871 411 25 22];
            app.CLabel.Text = 'ºC';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)
```

```
        % Construct app
        function app = app3

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```